ABSTRACT

Title of dissertation: ADAPTIVE ALGORITHMS FOR AUTOMATED PROCESSING OF DOCUMENT IMAGES

Mudit Agrawal, Doctor of Philosophy, 2011

Dissertation directed by: Professor Larry Davis
Department of Computer Science

Dr. David Doermann
University of Maryland Institute for Advanced Computer Studies

ABSTRACT

Large scale document digitization projects continue to motivate interesting document understanding technologies such as script and language identification, page classification, segmentation and enhancement. Typically, however, solutions are still limited to narrow domains or regular formats such as books, forms, articles or letters and operate best on clean documents scanned in a controlled environment. More general collections of heterogeneous documents challenge the basic assumptions of state-of-the-art technology regarding quality, script, content and layout. Our work explores the use of adaptive algorithms for the automated analysis of noisy and complex document collections.

We first propose, implement and evaluate an adaptive clutter detection and removal technique for complex binary documents. Our distance transform

Report Documentation Page

Form Approved
OMB No. 0704-0188

1. REPORT DATE	2. REPORT TYPE	3. DATES COVERED
2011		**00-00-2011 to 00-00-2011**

4. TITLE AND SUBTITLE	5a. CONTRACT NUMBER
Adaptive Algorithms for Automated Processing of Document Images	5b. GRANT NUMBER
	5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S)	5d. PROJECT NUMBER
	5e. TASK NUMBER
	5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) **University of Maryland, College Park,Department of Computer Science,College Park,MD,20742**	8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)	10. SPONSOR/MONITOR'S ACRONYM(S)
	11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT

Approved for public release; distribution unlimited

13. SUPPLEMENTARY NOTES

14. ABSTRACT

Large scale document digitization projects continue to motivate interesting document understanding technologies such as script and language identification page classification, segmentation and enhancement. Typically, however, solutions are still limited to narrow domains or regular formats such as books, forms articles or letters and operate best on clean documents scanned in a controlled environment. More general collections of heterogeneous documents challenge the basic assumptions of state-of-the-art technology regarding quality, script, content and layout. Our work explores the use of adaptive algorithms for the automated analysis of noisy and complex document collections. We first propose, implement and evaluate an adaptive clutter detection and removal technique for complex binary documents. Our distance transform based technique aims to remove irregular and independent unwanted foreground content while leaving text content untouched. The novelty of this approach is in its determination of best approximation to clutter-content boundary with text like structures. Second, we describe a page segmentation technique called Voronoi++ for complex layouts which builds upon the state-of-the-art method proposed by Kise [46]. Our approach does not assume structured text zones and is designed to handle multi-lingual text in both handwritten and printed form. Voronoi++ is a dynamically adaptive and contextually aware approach that considers components? separation features combined with Docstrum [64] based angular and neighborhood features to form provisional zone hypotheses. These provisional zones are then verified based on the context built from local separation and highlevel content features. Finally, our research proposes a generic model to segment and to recognize characters for any complex syllabic or non-syllabic script, using font-models. This concept is based on the fact that font files contain all the information necessary to render text and thus a model for how to decompose them. Instead of scriptspecific routines, this work is a step towards a generic character and recognition scheme for both Latin and non-Latin scripts.

15. SUBJECT TERMS

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT **unclassified**	b. ABSTRACT **unclassified**	c. THIS PAGE **unclassified**	**Same as Report (SAR)**	**194**	

Standard Form 298 (Rev. 8-98)
Prescribed by ANSI Std Z39-18

based technique aims to remove irregular and independent unwanted foreground content while leaving text content untouched. The novelty of this approach is in its determination of best approximation to clutter-content boundary with text like structures.

Second, we describe a page segmentation technique called Voronoi++ for complex layouts which builds upon the state-of-the-art method proposed by Kise [46]. Our approach does not assume structured text zones and is designed to handle multi-lingual text in both handwritten and printed form. Voronoi++ is a dynamically adaptive and contextually aware approach that considers components' separation features combined with Docstrum [64] based angular and neighborhood features to form provisional zone hypotheses. These provisional zones are then verified based on the context built from local separation and high-level content features.

Finally, our research proposes a generic model to segment and to recognize characters for any complex syllabic or non-syllabic script, using font-models. This concept is based on the fact that font files contain all the information necessary to render text and thus a model for how to decompose them. Instead of script-specific routines, this work is a step towards a generic character and recognition scheme for both Latin and non-Latin scripts.

ADAPTIVE ALGORITHMS FOR AUTOMATED PROCESSING OF DOCUMENT IMAGES

by

Mudit Agrawal

Dissertation submitted to the Faculty of the Graduate School of the
University of Maryland, College Park in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
2011

Advisory Committee:
Professor Larry Davis, Chair
Dr. David Doermann, Co-chair
Professor Ramani Duraiswami
Professor David Jacobs
Professor Min Wu
Dr. Wael Abd-Almageed

Dedication

I dedicate this to my grandfather, Shri S.P. Gupta, who couldn't be here to see the fruits of this day.

Acknowledgments

I express my heartfelt gratitude to all the people who have helped me make this thesis possible and because of whom my graduate experience has been one that I will cherish forever.

I thank God for giving me strength and motivation to pursue it with zeal and enthusiasm and for all the goodness that came along with it.

First and foremost I'd like to thank my advisor, Dr. David Doermann, for being extremely supportive and helpful throughout these years. His enthusiasm, work ethics, passion, knowledge and maturity in this field have been the most important elements of inspiration for me throughout this journey and I have always looked upon him. I would also like to thank him for giving me an invaluable opportunity to work on challenging and extremely interesting projects over the past years. He has always made himself available for help and advice and there has never been an occasion when I've knocked on his door, emailed him, called him on his mobile, pinged him on skype and he hasn't given me time. It has been a pleasure to work with and learn from such an extraordinary personality.

I would also like to thank my committee chair, Professor Larry Davis. Without his extraordinary expertise, this thesis would have been a distant dream. Thanks are due to Professor David Jacobs, Professor Ramani Duraiswami, Professor Min Wu and Dr. Wael Abd-Almageed for agreeing to serve on my thesis committee and for sparing their invaluable time reviewing my manuscript. I'd

also like to thank Dr. Stefan Jaeger for his support and ideas in the beginning phase of my Ph.D.

My colleagues at the language and media processing (LAMP) lab have enriched my graduate life in many ways and deserve a special mention. Elena Zotkina helped with some of the evaluations and tools. My interactions with Huanfeng Ma, Yi Li, Sameer Kibey, Rajat Ahuja, Guangyu Zhu and Jayant Kumar have been very fruitful.

I would also like to acknowledge the CS Graduate Program Coordinators, Jennifer Story and Fatima Bangura, who apart from being an excellent support for all paper-work and questions, are like my family at Maryland. Special thanks to our Administrative Assistant Janice Perrone for helping me over tight deadlines and conference trips. I would also like to acknowledge help and support from all of the UMIACS staff members.

My special thanks to my managers at Microsoft Corp., Fei Su and Sashi Raghupathy, who gave me an invaluable opportunity to always work on projects related to my Ph.D. at Microsoft and gave me those occasional breaks off-work to concentrate on my education.

I owe my deepest thanks to my family - my mother and father who have always stood by me and guided me through my career, and have pulled me through against impossible odds at times. Words cannot express the gratitude I owe them. I would also like to thank my brother and my grandmother for always being there for me when I needed them.

My friends at College Park, Maryland and Redmond, Washington have

been a crucial factor in the smooth completion of my thesis and have been very encouraging and motivating. I could not have accomplished this thesis without concessions from outings with my friends, even after I had cheated myself of the indispensable sleep time! I'd like to express my gratitude to all of them for their friendship, energy and support.

The financial support from the DARPA through BBN and US Government through NSF, for all the projects discussed herein, is gratefully acknowledged.

My thesis has been a collection of so many thoughts and efforts from so many people, that it seems a little tough to remember each one and put them on paper. I am deeply and truly thankful to all and even those who I might have inadvertently missed mentioning here.

Thank you all!

Table of Contents

List of Tables ix

List of Figures x

List of Abbreviations xv

1 Introduction 1
- 1.1 Noise Detection and Removal . . . 5
  - 1.1.1 Problem Definition and Challenges . . . 5
  - 1.1.2 Contributions . . . 8
- 1.2 Page Segmentation and Zone Classification . . . 9
  - 1.2.1 Problem Definition and Challenges . . . 9
    - 1.2.1.1 Page Segmentation . . . 11
    - 1.2.1.2 Zone classification . . . 11
  - 1.2.2 Contributions . . . 13
- 1.3 Character Segmentation and Recognition . . . 15
  - 1.3.1 Problem Definition and Challenges . . . 15
    - 1.3.1.1 Segmentation complexities . . . 16
    - 1.3.1.2 Limited availability of annotated data . . . 19
  - 1.3.2 Contributions . . . 20
- 1.4 Organization . . . 20

2 Noise Detection and Removal 23
- 2.1 Background and State-of-the-art . . . 23
  - 2.1.1 Clutter . . . 23
  - 2.1.2 Stroke-like Pattern Noise . . . 31
- 2.2 Clutter . . . 35
  - 2.2.1 Clutter Detection and Removal . . . 35
    - 2.2.1.1 Background: Distance Transform approach . . . 36
    - 2.2.1.2 Pre-processing . . . 38
    - 2.2.1.3 Clutter Detection . . . 39
    - 2.2.1.4 Clutter Removal . . . 42
  - 2.2.2 Evaluation . . . 49
    - 2.2.2.1 Datasets . . . 49
    - 2.2.2.2 Pixel-based Evaluation . . . 51
    - 2.2.2.3 Purposive Evaluation . . . 56
    - 2.2.2.4 Error Analysis . . . 57
- 2.3 Stroke-like Pattern Noise . . . 60
  - 2.3.1 Prominent Text Components . . . 60
  - 2.3.2 Noise Removal using a Content Understanding Technique . 61
    - 2.3.2.1 Supervised Prominent Text Component Classification . . . 61
    - 2.3.2.2 Unsupervised Small-Component Classification . . . 62

2.3.3 Evaluation . . . 65
2.3.3.1 Datasets . . . 65
2.3.3.2 Metrics . . . 66
2.3.3.3 Results . . . 66

3 Page Segmentation and Zone Classification 69
3.1 Background and State-of-the-art . . . 69
3.1.1 Area Voronoi Based Segmentation . . . 72
3.1.1.1 Algorithm . . . 72
3.1.1.2 Tuning parameters . . . 74
3.1.2 Docstrum Segmentation . . . 76
3.1.3 State-of-the-art Limitations . . . 78
3.2 Page Segmentation . . . 80
3.2.1 Voronoi++ Approach . . . 80
3.2.2 Hypothesis Phase . . . 81
3.2.2.1 Dynamic Adaptation . . . 82
3.2.2.2 Provisional Edges . . . 92
3.2.3 Validation Phase: Considering Context . . . 98
3.2.3.1 Separation Context . . . 99
3.2.3.2 Similarity Context . . . 99
3.2.4 Evaluation . . . 102
3.2.4.1 Metrics . . . 102
3.2.4.2 Datasets . . . 107
3.2.4.3 Experiments and Results . . . 109
3.2.4.4 Error Analysis . . . 117
3.3 Zone Classification . . . 122
3.3.1 Feature Extraction . . . 122
3.3.1.1 Structural Features . . . 123
3.3.1.2 Texture Features . . . 125
3.3.2 Classification . . . 127
3.3.2.1 Dimensionality Reduction using PLS . . . 127
3.3.2.2 Hybrid Multi-Class Classification . . . 128
3.3.3 Evaluation . . . 130

4 Character Segmentation and Recognition 131
4.1 Background and State-of-the-art . . . 131
4.2 Base Recognition System . . . 132
4.2.1 Segmentation . . . 132
4.2.2 Feature Extraction . . . 134
4.2.3 Classification . . . 136
4.2.3.1 Template Matching . . . 137
4.2.3.2 Nearest Neighbor Classifier and Weighted Euclidean Distance . . . 137
4.2.3.3 Hierarchical Classification . . . 138
4.2.4 Additional Challenges . . . 139

4.3 Font based Intelligent Character Segmentation 139
4.3.1 Benefits and Font Models 139
4.3.2 Training Using Font-files 142
4.3.3 Segmentation and Recognition 143
4.4 Evaluation 147
4.4.1 Datasets 147
4.4.2 Metrics and Tools 148
4.4.3 Experiments and Results 149
4.4.3.1 Feature Extraction 149
4.4.3.2 Character Segmentation 150
4.4.3.3 Character and Word Recognition 150

5 Summary and Future Directions 153
5.1 Noise Detection and Removal 153
5.2 Page Segmentation and Zone classification 159
5.3 Character Segmentation and Recognition 160

A Summary of Contributions 164

B List of Publications 166

Bibliography 168

List of Tables

1.1 Comparison of 4 scripts based on their properties 19

2.1 State-of-the-art algorithms for clutter removal. 30
2.2 Differentiating properties of residual images of clean and clutter document. 42
2.3 Detection Accuracy. 52
2.4 Removal Accuracy. 52
2.5 Purposive evaluation of clutter on line segmentation algorithm. . . 57

3.1 Evaluation Schemes 107
3.2 Datasets 108
3.3 Illustration of results of various approaches on handwritten Arabic dataset. 111
3.4 Performance Comparison 130

4.1 Compares character level accuracy results for Latin and Khmer scripts using Template and Directional Element Features (DEF). . . 150
4.2 Compares dissection and font-model based techniques. 152

List of Figures

1.1 Modules of a document analysis system. 5
1.2 (a) Document image showing clutter connected with ruled-lines and text. (b) Single connected component of clutter (c) Output image after noise detection and removal. 9
1.3 Comparing various algorithms based on the component features they use for segmentation. 12
1.4 Page Segmentation (a) using Voronoi (b) using Voronoi++. 14
1.5 (a) Syllable (b) Conjunct for Cambodian script. 18
1.6 (a) Burmese script (b) Devanagari character composition (c) Arabic Word rendering. 19
1.7 (a) Linear segmentation in Latin script (b) Complex bidirectional segmentation in Cambodian script (c) Successive horizontal and vertical segmentations in Indic scripts. 21

2.1 Examples of clutter. 24
2.2 (a) Clutter Image (b) Image showing a single connected component with text and clutter. 25
2.3 Dependent noise left after inaccurate clutter removal. 26
2.4 The Figure shows three lines on clutter. The first is the desired or actual boundary of separation between clutter and content. The second is the boundary of the clutter after erosion. The third is the boundary obtained after enlarging the eroded clutter up to the first encountered background pixel. 27
2.5 Figure depicts restrictive flood fill algorithm from [15]. α shows the stroke-width threshold, while β shows the font-size threshold. These thresholds help in determining if flood-fill is encroaching content-pixels. 29
2.6 Stroke-like Pattern Noise, resembling diacritics, present around text components. 32
2.7 Examples of Stroke-like Pattern Noise. 33
2.8 Clutter Detection and Removal Flowchart. 36
2.9 (a) shows a clutter with fragmented boundary (b) shows a clutter with smooth boundary with the background. 39
2.10 Figure shows foreground distance transform values along a path from center of the clutter to its boundary in case of (a) a fragmented boundary (b) smooth boundary. 40
2.11 (a) An image with clutter and text. (b) The distance transform on the image with distances normalized to gray values [0-255] (c) Result of residual process. 41
2.12 Clutter Removal Process. 44
2.12 Clutter Removal Process. 45
2.12 Clutter Removal Process. 46

2.13 Frequency graph showing a sharp rise at ρ. 47
2.14 Clutter Detection and Removal. 48
2.15 Pixel Distribution. 52
2.16 Results of our clutter removal approach: The left side shows the cluttered images and the right side shows the corresponding cleaned images as a result of our clutter removal approach. 53
2.16 Results of our clutter removal approach: The left side shows the cluttered images and the right side shows the cleaned images as a result of our clutter removal approach. 54
2.17 (a) Clutter document image (b) One big connected component containing clutter and majority of content (c) Clutter removed with text preserved. 55
2.18 Performance varies linearly with image size and foreground content. 55
2.19 Line segmentation results on clutter and cleaned images. 58
2.20 (a) Text-line image of spiral bindings at the top of the document image are treated as text (b) Clutter removal fails at removing text-line protrusions. 59
2.21 Stroke-like Pattern Noise is left behind after complete clutter removal process. 59
2.22 Red (dark gray) and black components depict PTCs and non-PTCs respectively. 60
2.23 (a) Stroke-like Pattern Noise, resembling diacritics, present around text components. (b) Image shows classified non-PTCs (smaller text and noise components) overlaid the distance transform map of PTCs. Components nearer the darker regions are closer to the PTCs and vice versa. 64
2.24 SPN removal result of the test image in Figure 2.6. The noise components are successfully removed. 65
2.25 Results of Stroke-like Pattern Noise Removal (a) Pixel Distribution (b) Accuracy. 67
2.26 Results of our SPN removal algorithm: The left side shows the noisy images and the right side shows the cleaned images as a result of our SPN removal approach. 68

3.1 Comparing various algorithms based on the component features they use for segmentation. 72
3.2 (a) Document image overlayed with Voronoi region of each component (b) Weak edges removed based on listed criteria, result is region segmentation [46]. 73
3.3 Figures show sensitivity of Voronoi based segmentation at various noise thresholds (a) 5 (b) 20 (c) 50 (d) 150. 75
3.4 Docstrum Segmentation [64], ©1993 IEEE (a) Part of original image (b) Nearest neighbor vectors overlaid on the image (c) Nearest neighbor vectors are shown. 77

3.5 Comparing state-of-the-art algorithms and their limitations for a context-aware feature-space. 79
3.6 The two-phased Voronoi++ approach. 82
3.7 Image showing over segmentation of large fonts and dissimilar text sizes grouped together. 84
3.8 Text-line direction can help merging the over-segmented regions as shown above. 85
3.9 Illustration of results with and without NN associativity: The left side shows segmentation results without nearest-neighbor associativity and the right side shows the corresponding results with nearest-neighbor associativity. 87
3.10 (a) Linear (b) Parabolic relation between inter-word distance and area ratio. 89
3.11 Frequency histogram of distances associated with edges. 90
3.12 (a) Page Segmentation using the original Voronoi approach (b) Page Segmentation after finding word separation thresholding using the improved valley-finding algorithm. 92
3.13 Dynamically Adaptive Voronoi assures both Voronoi and Docstrum features. 93
3.14 A and B depict increase in T_{d2} based on larger components, due to which dissimilar zones have been merged together. C depicts large components formed due to touching-characters *(be, Dec)*. This in turn increases T_{d2} and removes the edge, leading to the fusion of two columns into a single zone. 95
3.15 Highlighted area shows edges with slightly larger distances than T_{d2} and may be equally likely to be removed. 96
3.16 Components belonging to different zones are merged if separated by less than the character-separation threshold. 97
3.17 Non-critical, critical and provisional edges separating two zones. . 99
3.18 (a) shows 4 feature-vectors spaced in 2-D. All vectors are separated from each other by nearly equal distances. (b) shows few similar feature-vectors to each vector. The clusters become apparent. 101
3.19 Context-aware phases of Voronoi++. 103
3.20 Result zone covering two distinct zones is not penalized using line-based evaluation. 105
3.21 3 metrics are used for comparing various page segmentation algorithms. The rectangular blocks show ground-truthed zones and result zones are depicted as polygons. The noise regions are not ground-truthed. 106
3.22 Accuracy comparison for Arabic and English datasets. FN = False Negatives, FP = False Positives, TP = True Positives 109
3.23 Illustration of results of various segmentation approaches on printed English dataset. 110

3.24 Metrics comparison for English datasets. Voronoi++ improves accuracy consistently. FN = False Negatives, FP = False Positives, TP = True Positives . . . 113
3.25 Number of zones returned from Voronoi++ are closest to those in ground-truth across different evaluation metrics. . . . 114
3.26 Voronoi++ performs best at a consistent noise threshold (=20) across Arabic and English datasets, whereas Voronoi's accuracy peaks at different thresholds for different datasets. . . . 114
3.27 Recall of all approaches against different evaluation schemes. . . . 115
3.28 Voronoi++ results on various datasets. . . . 115
3.28 Voronoi++ results on various datasets. . . . 116
3.29 Faulty segmentation on noisy documents. . . . 118
3.30 (a), (b) Ambiguous and (c) incorrect ground-truthing for component-based segmentation. . . . 119
3.31 Incorrect segmentation for Tables and Charts. (a) Voronoi++ groups similar-looking parts, but tends to group spatially separated columns into zones. (b) Voronoi++ groups similar-looking parts, but tends to separate highly dissimilar parts of an image from the rest. . . . 120
3.32 (a) Smaller-sized punctuations around capitalized abbreviations tend to make Voronoi++ over-sensitive for certain regions. (b) Over-segmentation of Mathematical Equations. . . . 120
3.33 (a) Separation context fails due to curved component (b) Similarity context treats the merged zones similar. . . . 121

4.1 Bigger boxes (dotted) demonstrate syllable-level segmentation while smaller (solid) ones show character-level segmentation for (a) Devanagari script (b) Cambodian script composition. . . . 133
4.2 Directional Element Feature Extraction. . . . 136
4.3 Chart shows locations of 3 different characters of Devanagari script using 3 structurally similar Devanagari font files. . . . 141
4.4 The figure above shows rendering of characters in a word in Khmer using font-models (a) Locating first character (b) Placing character into its bounding-box (c) Determination of the origin of next character (d) Determination of next character's bounding-box (e) Placing the second character. . . . 142
4.5 Character training using font-files. . . . 144
4.6 Dynamic Network created during best-path search of word-recognition (using Font-models). . . . 146
4.7 Conjunct Detection. Bigger (dotted) Box shows a possible conjunct under detection. If two characters (solid boxes) fit in (using font-model), it may be a conjunct. . . . 147
4.8 (a) Sample text from English dataset (b) Sample text from Khmer dataset. . . . 149

4.9 Improvements of our technique over older dissection based techniques (a) and (b) show Latin script character segmentation using dissection font-model based techniques respectively (c) and (d) show results for Khmer script using dissection and our font-model based techniques respectively. 151
4.10 Character recognition error rates continue to drop using better feature extraction methods and an intelligent character segmentation system using font-files. 152

5.1 Noise Removal and Voronoi++ Illustration. 155
5.1 Noise Removal and Voronoi++ Illustration. 156
5.2 Documents with ruled lines touching character components. 158

List of Abbreviations

UMIACS	University of Maryland Institute for Advanced Computer Studies
NSF	National Science Foundation
DARPA	Defense Advanced Research Projects Agency

Chapter 1

Introduction

Recent advances in search technology, decreases in storage costs, and a general improvement of information access through the Internet, have rekindled interest in the mass digitization of existing document repositories for access purposes. Historically, large scale digitization projects have focused on special collections and have invested considerable effort in providing extensive and accurate metadata, with relatively little reliance on automated processing. Automated document processing has been reserved primarily for conversion tasks where the goal is to completely convert hardcopy to an alternative electronic representation of the original source. Typically these automated techniques are the most successful for known domains or structured page formats such as forms, articles or letters, although considerable progress is being made in the processing of heterogeneous collections.

There are a number of large scale digitization projects that continue to advance the field [1, 2, 3, 4, 5] and drive interesting triage problems such as script and language identification, page classification, segmentation and enhancement. They are however still typically focused on a known class of documents such as newspapers, census records or historical books. The Google Books project, for example, is only focused on published books, and are scanned in a fairly con-

trolled environment [2]. An alternative class of problems involves access to more general heterogeneous collections, where a much more diverse set of documents are captured, and perhaps with much less regard for quality. Typical applications may include litigation support, large scale capture of collections for triage and storage or simply document image management in office environments.

With the goal of accessing collections that may be many orders of magnitude larger than current digital libraries, the document analysis community will need to continue to focus on a number of difficult problems that arise from these more diverse, lower quality images. These collections challenge some of the basic assumptions on which state-of-the-art algorithms are based, including reasonable quality, consistent, structured layout and known script or language.

The first challenge is the quality of document images being produced as a result of scanning. Mass data scanning and the transmission of document images through low-bandwidth networks for storage and access may require high speed scanning and data reduction in the form of reduced resolution, reduced image depth and image compression. High speed scanners often produce artifacts such as page borders, skew or intensity variations. Many images produced are scanned as binary or automatically thresholded and saved as binary images, instead of gray-level or color. Generic thresholding often amplifies problems in subsequent phases, introduces various forms of noise, allows background pattern to flow into foreground contents, exacerbates touching and broken characters and leads to an overall degradation in document quality. State-of-the-art automated analysis assumes noise to be 'separable' from content in one or more feature spaces, like

intensity (for bleed-through), boundary (marginal noise) or sparseness (for salt-and-pepper). Techniques like region growing and morphological analysis have proved to be sufficient for noise removal for only in the latter cases. In this thesis, we will consider noise types in binary documents which tend to directly affect text in the foreground in irregular ways. The challenge is to detach and preserve text and eventually remove noise from the document.

The second challenge involves mixed content and layout. Handwritten documents are becoming increasingly visable in the document analysis community as progress continues to be made. Such free-form handwritten documents along with handwriting-annotated printed text documents are challenging the assumption of structured layouts and separable text regions. Non-Manhattan layouts resulting from such documents are far more complex than business letters or journal pages. The advancement of desktop publishing introduces additional challenges of layout analysis and text zone extraction from figures, diagrams, tables and other non-text regions. Rule-lines, local skew at the word or character level and different text-orientations within the same document lead to complexities in page segmentation and text-line extraction.

The third challenge arises from a desire to process and access content on a global scale. Commercial and state-of-the-art OCR systems deal with each script independently using script-specific character segmentation followed by script-dependent recognition routines. Each new script brings along unique challenges, making the vision of a generic Optical Character Recognition harder. It is very difficult to adapt the system to an unknown script, that may vary over time,

without knowing the details of its character-set composition and grammar. On the other hand, scripts of some less-commonly-taught or low density languages are often not addressed by commercial vendors for economic reasons. Limited amounts of annotated data and complex interaction between strokes in these scripts make the training of new classifiers challenging, prompting new ways to bootstrap systems to deal with them.

A typical document analysis work-flow is depicted in Figure 1.1 and covers two phases of our research. In phase one, a noisy multi-lingual document containing both machine-print and handwritten text is processed to identify zones of uniform content. Typically a document goes through preprocessing including normalization, enhancement, noise removal and a zone-extraction process where it is segmented into zones and each zone is classified by its type. Content-specific applications are then applied to each zone, e.g. the text zones are then typically passed to recognition. In the second phase, a character training model is built using ground-truth data of a given script. These models are then used to recognize appropriate content.

In this thesis, we focus on fundamental research topics in Noise Detection and Removal, Page Segmentation and Zone Classification as well as the Character Segmentation and Recognition for low density languages.

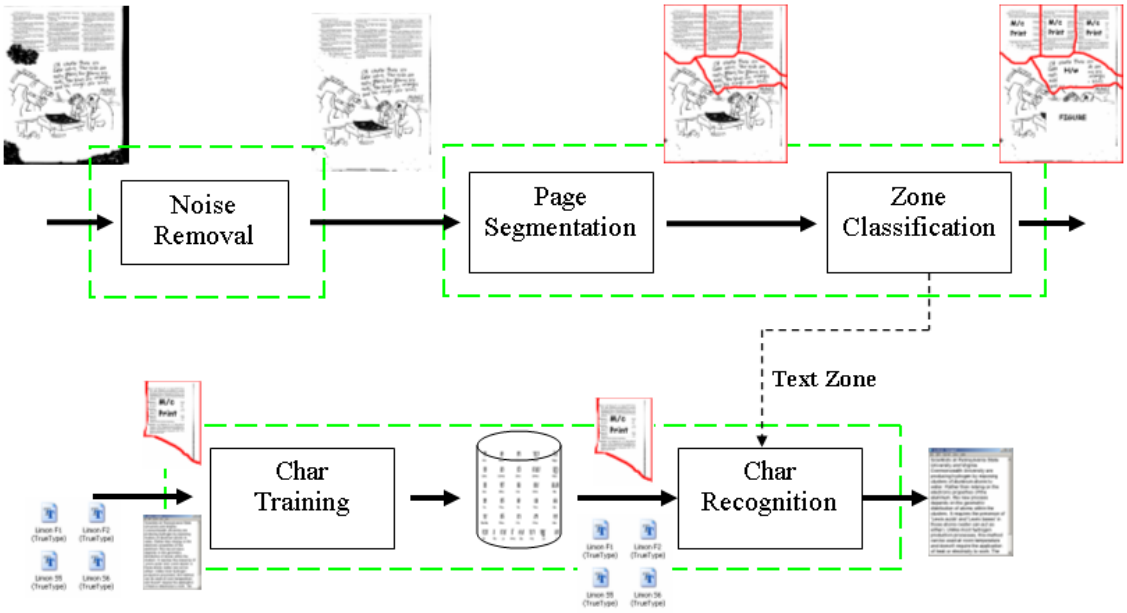


Figure 1.1: Modules of a document analysis system.

1.1 Noise Detection and Removal

1.1.1 Problem Definition and Challenges

Observed images often deviate from the ideal images that were produced by the source. These deviations may manifest themselves on the physical document in or after production or during scanning, transmission, storage or conversion from one form to another and are collectively refered to as noise. Irrelevant content, such as rule-lines, can also be viewed as noise, making the problem of detection and removal very much application dependent.

Document analysis algorithms such as page segmentation and character recognition typically work best on clean documents and often rely on connected

components as basic units, which unfortunately are sensitive to various types of noise.

Types of Noise: Document noise such as rule-lines [82, 9], bleed-through [92], stray-marks or clutter [15, 10] may be present before the scanning process, while many other types of document noise are introduced at later stages.

Clutter noise [10, 33] may also appear during the scanning process, due to the improper alignment of the document paper with the scanner bed or due to generic thresholding applied after scanning process. Similarly, bleed-through can also appear during scanning for thin paper and as a result of light reflecting off the scanner's backing.

Salt-n-pepper has been one of the most prevalent kind of noise in document images. Also known as bipolar noise, it is an impulsive noise which appears as randomly distributed small components over an image formed due to dithering binarization [22] and can be composed of one or more pixels. However, by definition, they have been assumed to be much smaller than the size of wanted content and, therefore, the most prominent techniques for removing salt-n-pepper noise use a small median filter [22, 87, 65], kFill window [73] or a morphological operator of size 3X3 or smaller [79].

Noise in binary document images can be viewed as dependent or independent of the underlying document content. Ink blobs, salt-n-pepper [28], stray marks and marginal noise [33] are, in general, independent of location, size or other properties of text data in the document image. Recorded images that have

this type of noise, can be expressed as the sum of true image $I(i, j)$ and the noise $N(i, j)$ as $R(i, j) = I(i, j) + N(i, j)$. Such noise can also be referred to as independent noise with respect to the content. On the other hand, when the noise is included inside the spatial frequency domain of the image and can not be suppressed without a priori knowledge of the content, it is referred to as dependent noise [94]. Blur, pixel-shift or bleed-through [92], manifest themselves differently depending on the content. Such dependent noise is comparatively more difficult to model, is mathematically non-linear and often multiplicative.

Document cleaning can be performed in two fundamental ways. One approach is to extract the information content out of an image, leaving non-content as noise [88], [99], [97] while the other approach is to detect and remove noise from an image, resulting in a clean document. The former approach is often used in cases where there is a limited number of content types. In particular, there has been a lot of research in extracting text from images [88], [89], [99], [97]. However, for varied content such as text, logos, figures, stamps, diagrams, equations, drawings or halftones, individual extraction processes may be required [70], [88], [104], [30], which makes this approach less practical. In addition, these individual extraction processes are typically dependent on zone layout analysis, which in turn are dependent on a clean document for good results.

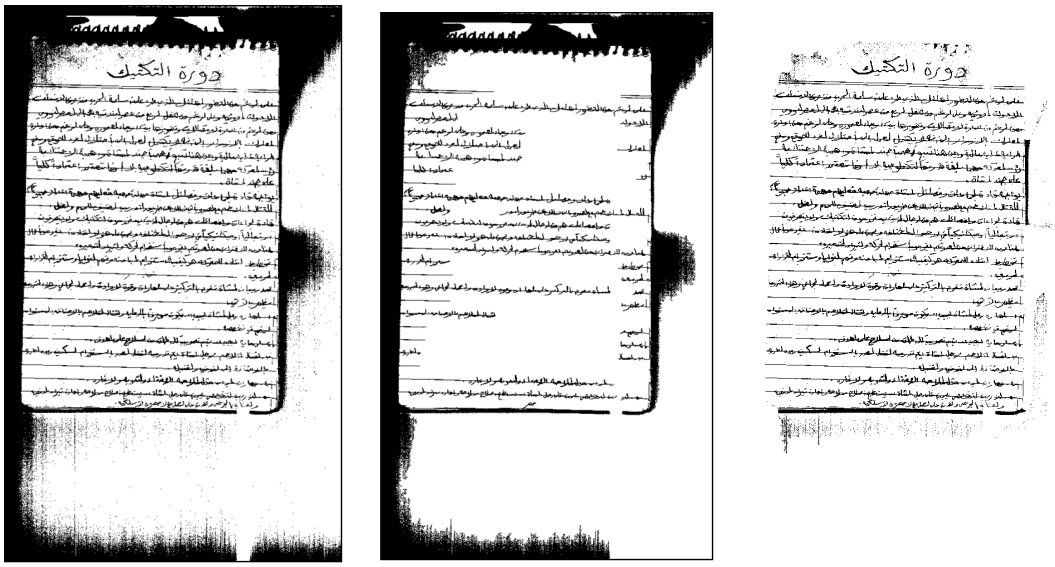
Detecting and removing noise is generally based on its properties like its shape, position, frequency, gray-values, density or periodicity of occurrence in the document. Unwanted punched holes exhibit regularity in their shapes, marginal noise [33] show regularity in their positions, while rule-lines [101], [103] show

periodicity in their positions and consistency in direction. On the other hand, noise such as ink blobs, complex background binarized patterns are denser than text, whereas salt-n-pepper [28, 12] is impulsive noise and is sparser than content pixels. If noise shows a consistent behavior in terms of these properties, it is easier to detect it and separate it from content. Ozawa and Nakagawa [68], Wang and Tang [92], Negishi et. al. [62] use gray-level to distinguish foreground from background. Fan et. al. [33] assumes length, position and neighborhood of noise to detect and remove it. Liang et al. [50] depend on periodicity of noise to get rid of it. However, there has been little work reported on the removal of noise which does not adhere to a consistent shape, position or size and which tends to interact with text in the foreground in irregular ways.

1.1.2 Contributions

In this thesis, we will consider noise in binary documents which appears as unwanted foreground content and tends to directly interact with text in irregular ways. We propose a novel technique for the identification and removal of unwanted foreground content in form of clutter and Stroke-like Pattern Noise (SPN). The challenge is to detach and preserve text and eventually remove noise from the document. Our detection and removal process is independent of noise position, size, shape and connectivity with text.

Figure 1.2a shows an image with clutter present around the handwritten text. Removal of the clutter's connected component (after detection) may lead to



(a) (b) (c)

Figure 1.2: (a) Document image showing clutter connected with ruled-lines and text. (b) Single connected component of clutter (c) Output image after noise detection and removal.

more than 70% of text-content removal as it is attached to the clutter in irregular fashion (Figure 1.2b). Morphological operations are known to degrade text. Our proposed pixel-based techniques remove clutter restrictively by preserving the text-content as shown in Figure 1.2c.

1.2 Page Segmentation and Zone Classification

1.2.1 Problem Definition and Challenges

With the advancements in information and communication technology and success in document understanding, there is a growing expectation that all forms of paper documents can be scanned, interpreted, indexed and treated as a legitimate form of media (like magnetic tapes and optical discs) which is both

machine and human readable [64]. This has led to the variety of paper documents scanned today being much more diverse than was typical several years ago. New scripts, more complex, non-Manhattan, computerized page layouts and various font styles are making this vision challenging. Furthermore, a larger percentage of handwritten material is being acquired which does not adhere to traditional layout constraints. Character recognition as well as established preprocessing modules such as noise removal, layout analysis and zone classification are effected negatively by this increase in complexity.

Another challenge involves mixed content and layout. Mixed handwritten and machine-printed documents are becoming increasingly visable in the document analysis community as progress continues to be made on processing them. Free-form handwritten documents along with handwriting-annotated printed text documents are challenging the assumption of structured layouts and separable text regions. Non-Manhattan layouts resulting from such documents are far more complex than business letters or journal pages. The advancement of desktop publishing introduces additional problems of layout analysis and text zone extraction from figures, diagrams, tables and other non-text regions. Ruled lines, local skew at the word or character level and different text-orientations within the same document lead to complexities in page segmentation and text-line extraction.

1.2.1.1 Page Segmentation

Current page segmentation algorithms make use of typical assumptions of uniformity in line, word and character spacing on a page. These assumptions fail however for handwritten and noisy documents where the size of components vary drastically due to cursive nature of handwriting, leading to over or under-segmentation. These algorithms lack the ability to dynamically adapt local variations in the size, orientation and distance of components within a page. Moreover, it is not necessary that components within a zone are either linearly separated or are of similar size. Instead, local patterns inside a zone are more likely to be consistent. At the same time, two adjacent regions of similar patterns can form distinct zones (e.g. two vertical columns on a page). Hence, a zone could be differentiated from its neighbors based on both its separation and its content. Voronoi based approaches [46] achieve this by using the area of components as a content-property. Clearly, area property is neither a necessary nor a sufficient condition for content differentiation and a more robust mechanism of computing a zone's pattern is required. Figure 3.1 compares state-of-the-art algorithms and the properties they use to make decisions on segmentation.

1.2.1.2 Zone classification

Identifying the content of document zones, that have previously been segmented, is a fundamental component of modern document analysis systems. For example, identifying a zone type allows the application of content-specific algo-

Algorithm	Separation: Distance	Separation: Direction	Content: Area Ratio
RLSA	✔	✔	
X-Y Cuts	✔		
Whitespace Analysis Method	✔	✔	
Constrained Text-line Detection	✔	✔	
Docstrum	✔	✔	
Voronoi	✔		✔

Figure 1.3: Comparing various algorithms based on the component features they use for segmentation.

rithms such as Optical Character Recognition (OCR). More importantly, zone type identification is crucial to indexing and retrieval of large document databases.

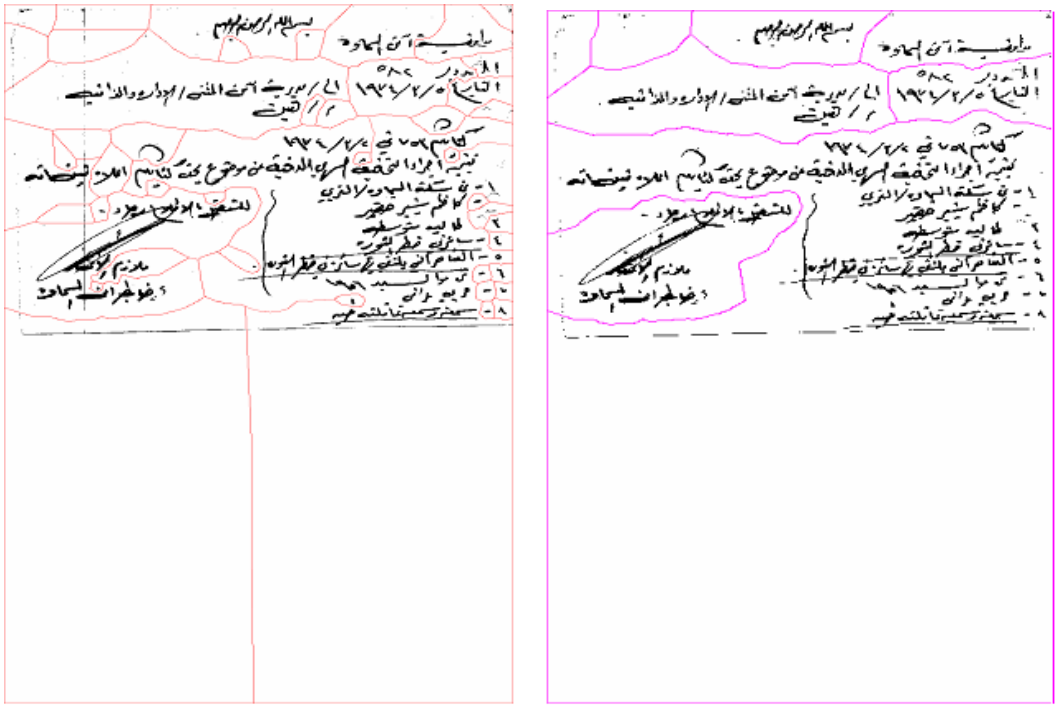
Broadly speaking, content analysis algorithms can be classified as one of three main approaches – (1) zone detection, (2) page classification or (3) zone classification .

Detection approaches, emphasize finding specific instances of zones, such as text regions [98], logos [26], mathematical expressions [100] or tables [41], without requiring segmentation of the page. Page classification approaches, assume the content of the entire page is of a single type (e.g. title page or index page) and a classifier is used to determine the page content [24][32]. Finally, zone classification approaches assume that the page is segmented into zones of independent content. Low level image features are extracted from each zone and a statistical classifier is used to label the different zones into one of possible content types (e.g. text, math, etc.) [93].

1.2.2 Contributions

In this thesis, we first develop a hybrid page segmentation approach which first estimates zone boundaries dynamically, then considers context to selectively prune them. The first *hypothesis* phase applies a deterministic approach for non-Manhattan layouts, which builds upon Kise et. al.'s work using Area Voronoi Diagrams [46]. Instead of linear relations between distance and area ratio of connected components, we dynamically determine these relations locally and consider angular and neighborhood features from Docstrum based features [64] to improve accuracy. The zone boundaries, however, are not removed unless sufficient evidence is found. Instead, low confidence boundaries are marked and kept for further validation. The second *validation* phase, introduces a contextually-aware improvement based on zone separation and content-type. This phase considers the low-level component relationship features from the hypothesis phase and high-level features based on the zone-content to evaluate a possible zone-merger using a semi-supervised clustering technique.

It improves page segmentation especially for handwritten documents, where the size of components vary drastically due to the cursive nature of writing, leading to over or under-segmentation using state-of-the-art techniques. Zone-based evaluations performed on sets of printed and handwritten documents, in English and Arabic scripts with multiple font types and sizes, show that we achieve an accuracy improvement by 74% over the accuracy reported in [46]. Figure 1.4 illustrates page segmentation results using Voronoi and Voronoi++.



(a) (b)

Figure 1.4: Page Segmentation (a) using Voronoi (b) using Voronoi++.

Second we propose a new combination scheme of structural and texture features to represent each region for zone classification. Partial Least Squares (PLS) [95] is then used to reduce the dimensionality of the feature space and find discriminating features. Rather than using the classic one-against-all or one-against-one approach for zone classification, a new hybrid approach seeking to improve the classification accuracy will be shown to give better results.

1.3 Character Segmentation and Recognition

1.3.1 Problem Definition and Challenges

There are many commercial omnifont, script dependent recognizers on the market, including ABBYY's FineReader, Nuance's OmniPage and Sakhr's Automatic Reader. Such recognizers generally make assumptions about the language, script and/or document quality, try to recognize a wide variety of fonts, sizes and script properties and are developed with extensive training with large numbers of training samples. Despite ongoing research on non-Latin script recognition, most of the commercial OCR systems focus on Latin based languages. Efforts on non-Latin scripts are quite focused and continue to be tailored for specific scripts using their inherent features explicitly. The resulting systems are often costly and do little to advance the field. Non-Latin scripts primarily pose two major challenges as compared to Latin scripts. The first challenge is the inherent complexity of the scripts in terms of recognizable glyphs and word compositions. This directly affects the segmentation of words into meaningful recognizable units, glyphs or characters. The second challenge is the unavailability of a large training set covering enough samples of each recognizable unit, especially for low-density languages.

1.3.1.1 Segmentation complexities

The literature often distinguishes between the recognition of scripts with isolated characters and scripts which are connected. With isolated scripts, characters are written to be separable (although they may touch due to degradation) while connected scripts can not be easily segmented. Casey and Lecolinet define four approaches to word recognition [23]. This covers scripts with isolated characters as well as connected scripts.

1. The first is dissection based where the word image is decomposed into classifiable units, glyphs or characters, before feature extraction and classification. Due to its disconnectivity from recognition module, any errors performed at the segmentation stage magnify recognition failures.

2. The second method classifies subsets of spatial features collected from a word image as a whole. Segmentation hypotheses are generated and choosing the best hypothesis along the word gives the best recognition result. This approach generally performs better than the previous one, due to the construction of a lattice with possible segmentations. However, the challenge is to come up with a minimal number of possibly correct hypotheses.

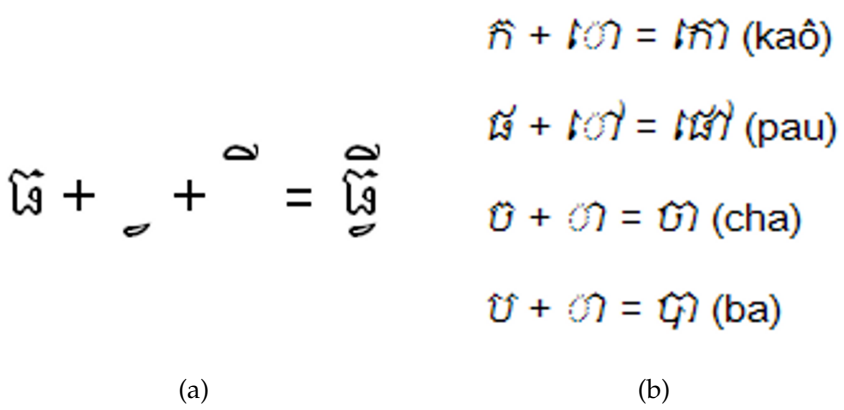
3. The third involves over-segmenting the word image using physical feature points. Although these techniques do fairly well in the handwriting domain, they have not yet been established in printed-character domain.

4. The fourth method recognizes an entire word as a unit and is a holistic strat-

egy. While the first three are segmentation based recognition approaches, this is a segmentation-free approach. Though this approach bypasses the complexities of character segmentation, it involves large amounts of training data and is limited to a predefined lexicon. This approach is generally not applied to languages like English, where a comparatively small character set gives rise to a much larger vocabulary. Therefore, for English, the ratio of number of classes required in a segmentation-free approach to that in segmentation-based approach is much larger than for other scripts.

Gader and Mohamed [59] proposed a combination scheme of segmentation-based and segmentation-free word recognition. The segmentation-free technique constructs a continuous density hidden Markov model for each lexicon string. The segmentation-based technique uses dynamic programming to match word images and strings. The combination module uses differences in classifier capabilities to achieve significantly better performance.

In addition to isolated vs. connected, scripts can also be broadly classified based on word-composition into *syllabic* and *non-syllabic*. In non-syllabic scripts, characters are horizontally (or vertically) separable glyphs whereas in syllabic scripts, glyphs appear as syllables, which are in turn a complex combination of one or more characters. Sometimes, characters fuse together to form new shapes, called ligatures (conjuncts) (Figure 1.5). Cambodian (Khmer) and many South-East Asian scripts are examples of syllabic scripts. The presence of language-specific constructs, in the domain of syllabic scripts, such as



(a) (b)

Figure 1.5: (a) Syllable (b) Conjunct for Cambodian script.

shirorekha (Devanagari), modifiers (South-East Asian scripts), writing order or irregular word-spacing (Arabic and Chinese) require different approaches to segmentation. Asian scripts, for example, share many common properties yet pose some unique challenges for segmentation. For example, in Burmese (Figure 1.6a), a vowel can be used with diacritics to create other vowels. Figure 1.6b shows a word in Devanagari where characters are combined together to form a word. A character's appearance is affected by its ordering with respect to other characters, the font used to render the character, and the application or system environment. Figure 1.6c shows the word *al-arabiyyah*, in Arabic, at three stages of rendering. The first line shows the individual letters, the second line shows it with the bidirectional display mechanism and the third line renders the letters using a glyph shaping mechanism according to context. While conventional vertical or horizontal profiling methods fail to segment characters directly from words, character segmentation from syllables using only connected component analysis itself is a complex task which is highly correlated with the script characteristics. Table 1.1 compares 4 scripts based on their features:

(a) (b) (c)

Figure 1.6: (a) Burmese script (b) Devanagari character composition (c) Arabic Word rendering.

Table 1.1: Comparison of 4 scripts based on their properties

Script	Continuous	Syllabic
English		
Khmer		√
Devanagari	√	√
Arabic	√	√

1.3.1.2 Limited availability of annotated data

In order to cover all the recognizable units (syllables, conjuncts or characters), systems typically need a much larger training set for syllabic scripts as compared to non-syllabic scripts. This is because syllabic scripts tend to have a larger character set and an even larger syllabic or conjunct set produced as a result of these atomic character-combinations. For example, Devanagari has 37 consonants and 16 vowels. These can appear in isolation as characters within a word or theoretically every consonant can combine with any vowel to form a syllable. In addition, two or more consonants can combine to form a conjunct, making the number of recognizable units much larger. Unfortunately, low-density languages

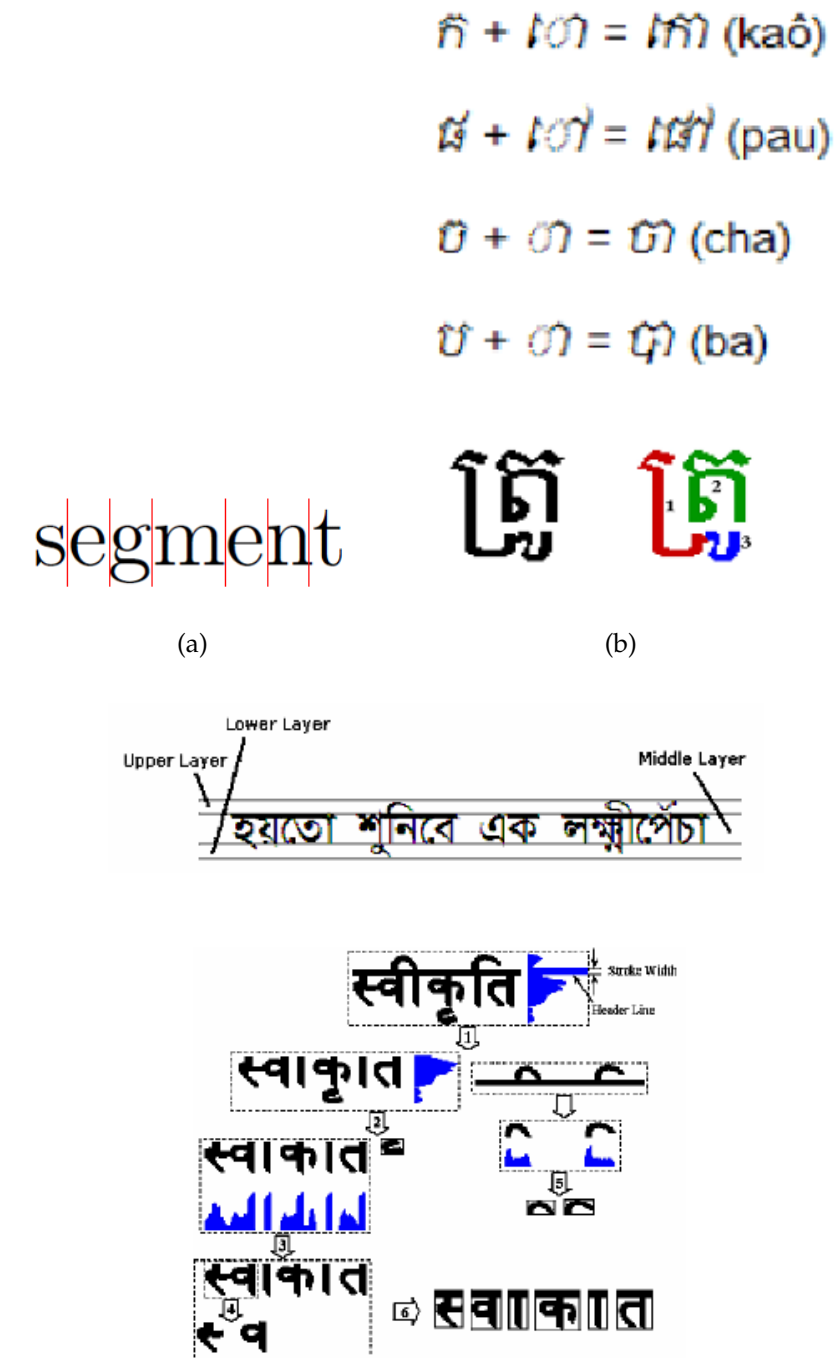
like Khmer, which have properties similar to South-East Asian scripts, come with very limited ground-truth data. Hence representation of every recognizable unit can not be guaranteed.

1.3.2 Contributions

In this thesis, we propose a generic character segmentation scheme using font files. The concept is based on the fact that font files contain all the information necessary to create the glyphs and thus a model for how to decompose them. Instead of script-specific segmentation routines, this work is a step towards a generic character segmentation and recognition scheme for both Latin and non-Latin scripts. Figure 1.7 shows three scripts with very different compositional rules. In Latin scripts, like English (Figure 1.7a), characters align themselves in linear fashion. In Indic scripts, like Bengali and Hindi (Figure 1.7c), a single word can be divided into three layers - upper, middle and lower, and vowels can combine with consonants in a bidirectional fashion. Cambodian script (Figure 1.7b) allows characters to combine in bidirectional fashion, without forming distinguishable layers. This work aims to create a generic framework to segment characters in such diverse scripts.

1.4 Organization

The thesis is organized as follows. Each module is described in one chapter along with its evaluation and results. Chapter 2 describes the noise detection and



(a) (b)

(c)

Figure 1.7: (a) Linear segmentation in Latin script (b) Complex bidirectional segmentation in Cambodian script (c) Successive horizontal and vertical segmentations in Indic scripts.

removal technique and page segmentation and zone classification is described in Chapter 3. This is followed by font-based character segmentation and recognition approach in Chapter 4. Chapter 5 summarizes the thesis and its contributions and discusses the possible additional work in these areas.

Chapter 2

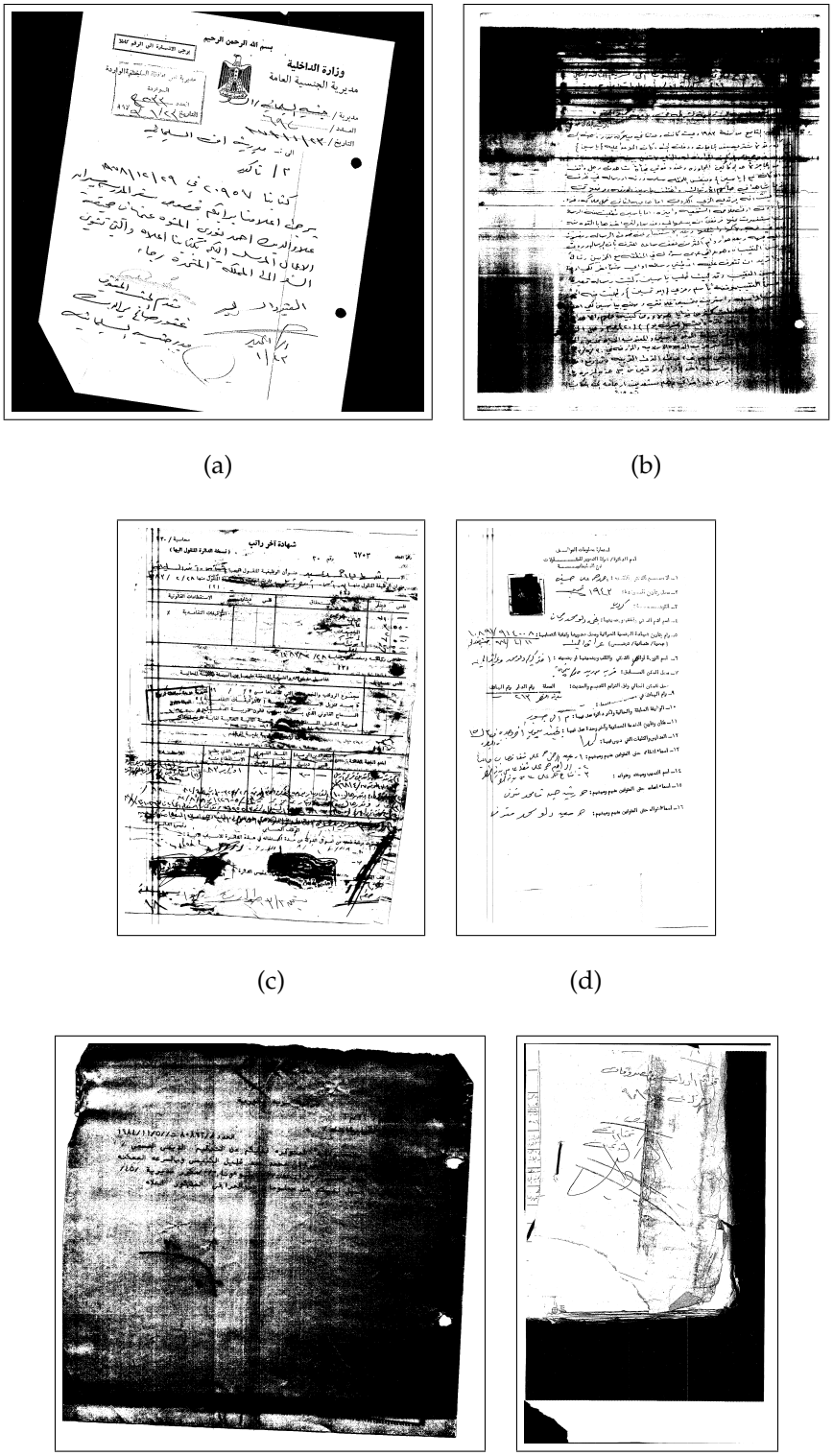
Noise Detection and Removal

2.1 Background and State-of-the-art

2.1.1 Clutter

In our work, clutter is used as a general term to refer to unwanted foreground content which is typically larger and much denser than text in binary images. It can result from numerous sources. While some forms of clutter like punched holes (Figure 2.1 a), ink seeps (Figure 2.1 b), ink blobs (Figure 2.1 c) and copier borders typically are present before the scanning process, other types of marginal noise may result from the scanning of bound or skewed documents (Figure 2.1 a, f for example), where the gap between the gutter and scanner or between edges of paper and scanner bed causes lighting variations. Other scanning and binarization artifacts may give rise to clutter as well (Figure 2.1 d, e). Clearly, clutter is predominantly text independent and irregular.

One of the major issues with clutter is its connectivity with text. Clutter often touches or overlaps some parts of the text. In the case of rule-line documents with clutter, a single connected component connecting clutter, ruled lines and text may be present (Figure 2.2). Complete removal of the connected component in such cases may result in tremendous loss of content while morphological



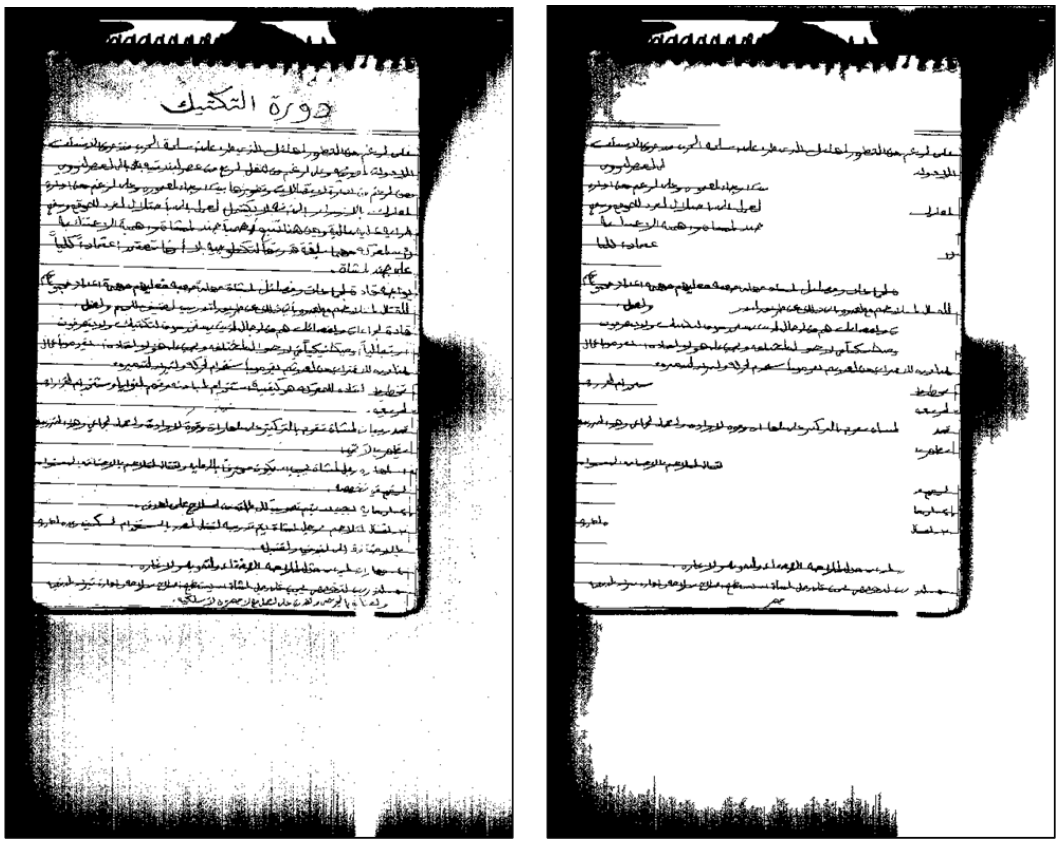
(a) (b)

(c) (d)

(e) (f)

Figure 2.1: Examples of clutter.

operations can degrade the text because the density of noise is much higher than that of the text. Our goal is to determine clutter-content boundary accurately so that the clutter can be fully removed while preserving the underlying text. While aggressive clutter removal may lead to content loss, conservative removal may leave traces of clutter behind, giving rise to dependent noise as shown in Figure 2.3.



(a) (b)

Figure 2.2: (a) Clutter Image (b) Image showing a single connected component with text and clutter.

As far as we know, there has been no collective work on the detection and removal of all forms of clutter, without removing or further degrading the attached text in binary document images. Fan, Wang and Kay [33] detect and

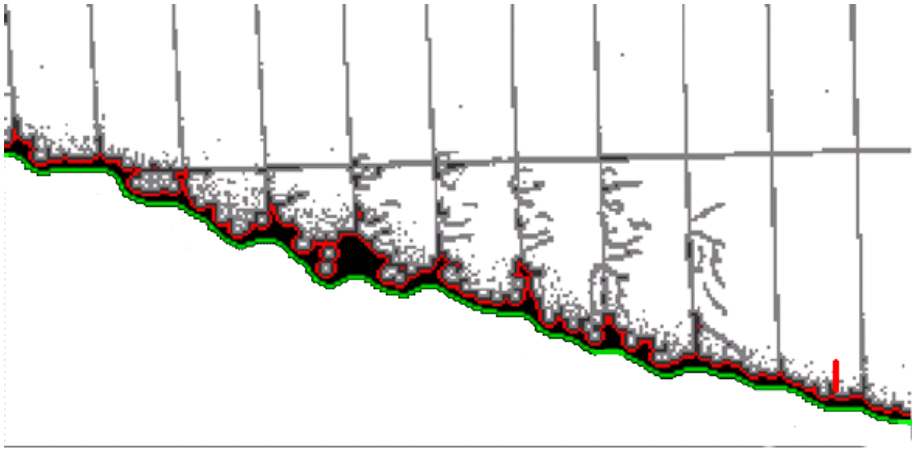


Figure 2.3: Dependent noise left after inaccurate clutter removal.

remove marginal noise regions based on three assumptions of shape, length and position. Image resolution is first reduced to get rid of text pixels, and regions of marginal noise are detected using shape, length and position requirements. The noisy region is then enclosed in an approximate rectangular shape which is then enlarged in consecutive steps until it encounters the first background pixel in the original resolution image. The enclosing portion of the polygon is the maximum portion of marginal noise that can be removed without removing the attached content. The technique does fairly well at removing only the marginal noise without the attached text but assumes a linear boundary of separation between marginal noise and content. Another problem is that the footprint or shape of clutter-boundary may be different from the one obtained after image down-sizing or morphological erosion.

Figure 2.4 illustrates the difference. Moving from the boundary towards the interior of the clutter, the first marked line is the boundary of maximum

erosion which erodes the text and still preserves the footprint of the clutter-boundary. Reduced clutter, if dilated by the same amount from this boundary, covers original clutter completely thereby separating the clutter from its attached text. The second line is the boundary of the clutter obtained after aggressive erosion or down-sizing the image. If we were to enlarge (dilate) this eroded clutter up to the first encountered background pixel, we will end up on third boundary. The removal of clutter up to this boundary will leave parts of clutter along with content. These chunks left behind become dependent on text and hence much more difficult to remove, as shown in Figure 2.3.

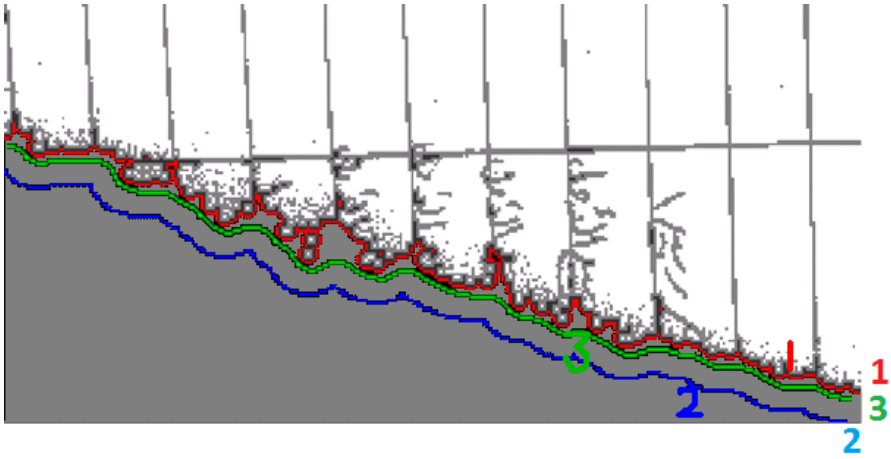


Figure 2.4: The Figure shows three lines on clutter. The first is the desired or actual boundary of separation between clutter and content. The second is the boundary of the clutter after erosion. The third is the boundary obtained after enlarging the eroded clutter up to the first encountered background pixel.

Shafait and Breuel [80] use black and white filters as well as positions of connected components to remove border noise from document images. Rectangular windows are moved in specific corner areas of a document image and the

percentage of black pixels is calculated. Whenever the percentage is above (black filter) or below (white filter) a given threshold, window-areas are classified as border noise or gutters between borders and text-regions respectively. Decisions to remove border noise is then based on the positions of these classified areas. Parameters like the rectangular window-size, thresholds for black and white filters, and the expected areas of border noise, make this approach dependent on a given resolution, font-size and specific to border-type clutter only. This approach also assumes a linear white-gutter boundary separating clutter and text-regions.

Stamatopoulous et al. [84, 85] rely on vertical and horizontal projection profiling methods to determine the linear boundary of separation between clutter (in form of border) and text. Unfortunately, any text attached to the border is removed in the process of noise removal. Le et al. [49] use rule-based classification of blank, textual and non-textual blocks based on the percentage of black pixels, locations, projection profiles and crossing counts. They also assume a linear white-gutter boundary separating clutter and text-regions, failure of which results in text-loss during noise removal. Both of these approaches rely on the average character heights determined from the document image based on the connected components. For noisy documents depicted in Figure 2.2, characters may not appear as individual components.

Avila and Lins [15] remove marginal noise regions by using a restrictive flood-fill algorithm. The basic algorithm moves outside in from the noisy surrounding border towards the document using a flood-fill technique. In order to restrain the flow into the content, a predefined stroke-width threshold (α) and

a predefined text-size threshold (β) are used. The idea is depicted in Figure 2.5. Although the algorithm tries to identify a non-linear boundary between noise and content, the predefined thresholds restrict its usage across diverse documents of varying resolution, scripts and font-sizes. Table 2.1 summaries the state-of-the-art algorithms, their basic technique and the assumptions they make towards clutter-noise removal.

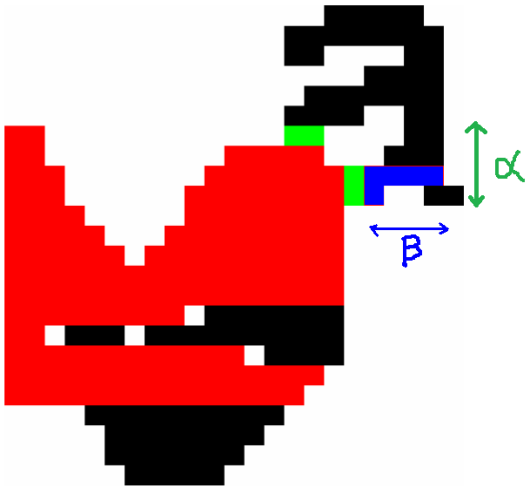


Figure 2.5: Figure depicts restrictive flood fill algorithm from [15]. α shows the stroke-width threshold, while β shows the font-size threshold. These thresholds help in determining if flood-fill is encroaching content-pixels.

In contrast, our technique aims at determining clutter-content boundary precisely on all forms of clutter. The advantages of our approach over the prior art are:

1. Position independence: Our approach is not limited to the marginal noise and attempts to clean clutter appearing at any position in the document.

2. Size independence: In accordance with the definition of clutter, as long as

Table 2.1: State-of-the-art algorithms for clutter removal.

Author(s)	**Noise Type**	**Technique**	**Assumptions and Limitations**
Fan, Wang and Kay [33]	Marginal Noise	Resolution reduction and polygonal enclosure of noisy borders based on margin's position and shape	Approximate determination of boundary of separation between clutter and text
Shafait and Breuel [80]	Textual and Non-textual border	Black and white filters and positions of connected components	Specific to a given resolution, font-size and border-type clutter only. Also assumes a linear white-gutter boundary separating clutter and text-regions
Stamatopoulous et al. [84, 85]	Textual and Non-textual border	Projection Profiling	Linear boundary of clutter-text separation. Attached text is removed during noise removal
Le et al. [49]	Marginal Noise	Rule-based classification of blank, textual and non-textual blocks based on the percentage of black pixels, locations, projection profiles and crossing counts	Linear white-gutter boundary separating clutter and text-regions
Avila and Lins [15]	Marginal Noise	Restrictive flood-fill	Pre-defined text-size and stroke-width thresholds

its size is larger than text, our technique is able to detect and remove it. In particular, we do make an assumption of clutter being at least two times larger than the text's stroke width (details in Section 2.2.1.3).

3. Shape independence: Unlike marginal noise, clutter can appear in various other forms, as illustrated in Figure 2.1. Our technique attempts detection and removal of all forms of clutter and is independent of the inclusion of any other type of noise.

4. Precise clutter-content boundary: Minimal dependent noise, in the form of clutter pixels close to the text, is left behind during clutter removal - defining a precise clutter boundary.

5. Minimal content degradation: Our intermediate step of residual image aims at locating and processing clutter components only. This does not degrade text in the document image.

2.1.2 Stroke-like Pattern Noise

Stroke-like Pattern Noise (SPN) [51] is of magnitude (size) similar to that of text-diacritics and tends to directly affect text in the foreground in irregular ways, as shown in Figure 2.6.

SPN is independent noise with respect to the content [10]. In general, it is independent of location, size or other properties of text data in the document image. In spite of being independent, due to its similarity to diacritics, its presence

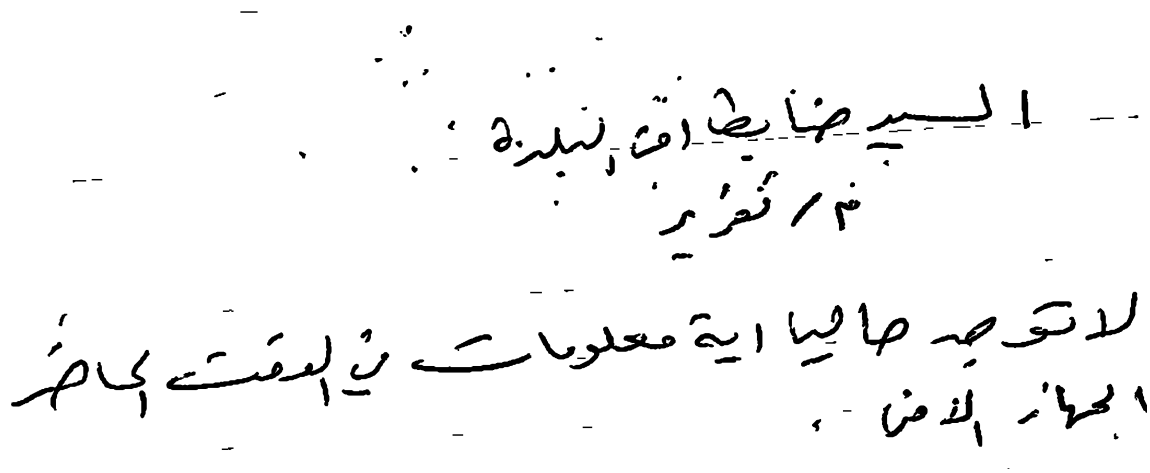
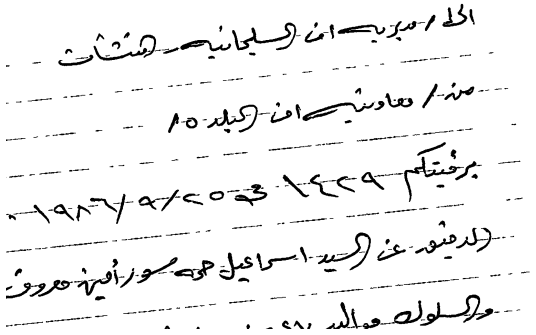


Figure 2.6: Stroke-like Pattern Noise, resembling diacritics, present around text components.

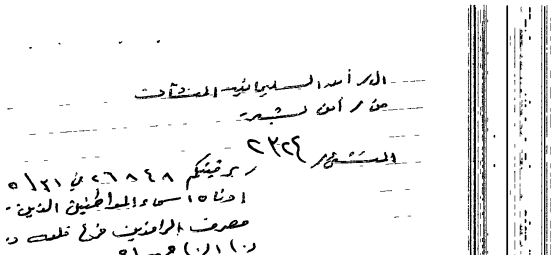
near textual components can change the meaning of a word, especially in Arabic documents.

This noise is formed primarily due to the degradation of underlying page rule lines that interfere with the foreground text. These degraded rule lines are severely broken, not straight and interact significantly with text. (Figure 2.7a). Another major source of formation are the blurring edges of clutter noise [10, 15] which remain after clutter removal approaches (Figure 2.7b). Stray marks in handwritten documents, some highly degraded and unperceivable background content can be other sources of such noise, as shown in Figures 2.7c, 2.7d.

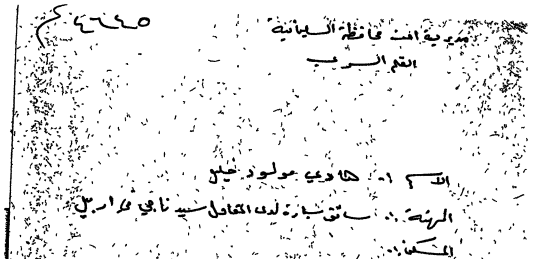
As degraded rule-lines, these line components are broken and degraded to a degree that they cannot be perceived in straight lines even by the human eye. This makes techniques like Hough transform, projection profiles not suited for their removal. Their shape and size similarity to smaller text components, prohibits morphological processing based removal approaches because the suc-



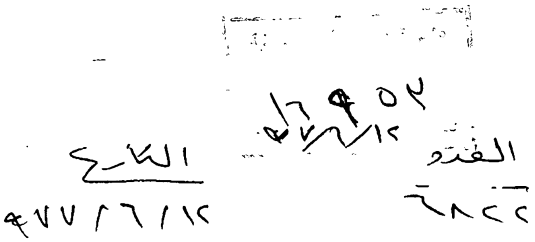
(a) Rule-line degradation



(b) Clutter residues



(c) Marks



(d) Degraded Background

Figure 2.7: Examples of Stroke-like Pattern Noise.

cessive erosion and dilation steps needed, tend to degrade text. Their similar spatial frequency to text renders median filtering approaches ineffective. With tremendous amount of research being done for salt-n-pepper noise and rule-line removal, this type of noise has thus been neglected as either aberrations or too degraded to model.

Document cleaning can be performed in two fundamental ways. One approach is to detect and remove noise from an image, and the other approach is to extract information content from the image, leaving non-content as noise behind. The former approach is prefered when the noise can be differentiated from text using their independent set of features. For example, clutter [10, 15], rule-lines [101, 103], salt-n-pepper noise [12, 73, 28] and marginal noise [33] exhibit properties quite different from the textual content. On the other hand, SPN cannot be removed without apriori knowledge of the textual content. This leads to the latter approach which aims at understanding content.

There has been a lot of work on extracting text components from a document image. However, the majority of the work has been focused on extracting text from colored documents or from background patterns. Using depth as an added dimension, all these algorithms benefit from gray-scale or color histogram analysis in order to differentiate text from background patterns [97, 83]. There has not been much work in differentiating handwritten (or printed) text in binary document images from stroke-like pattern noise (SPN).

Classifying all the text components and SPN in one step using a binary classifier entails using an extensive set of features capturing both shape and context

information at component-level. Apart from generating a detailed feature-set, this approach suffers from script-specific associations of smaller text components to the bigger ones. In order to cover all the recognizable units, across scripts, systems need a much larger training set. Limited amounts of annotated data at pixel level for many low density languages and complex interaction between strokes prompt for new ways to bootstrap systems to perform similar tasks.

This chapter is organized into two sections, one each for Clutter and Stroke-like Pattern Noise. Section 2.2 describes the problem definition, clutter detection and removal. Section 3 extends this approach to propose a generic noise removal model in which several forms of noise can be removed iteratively, without interfering in the detection and removal of clutter. This is followed by experiments and evaluation in Section 2.2.2. Section 2.3.1 outlines the proposed content understanding approach for Stroke-like Pattern Noise and Section 2.3.2 describes the two phases of the proposed approach. This is followed by evaluation in Section 2.3.3.

2.2 Clutter

2.2.1 Clutter Detection and Removal

Distance transforms are generic and accurate and the map obtained after distance transforms can be used to perform image analysis in two passes, unlike morphology which is recursive. Given an image, our first task is to determine if it contains clutter. For efficiency, clean documents should bypass the noise removal

process. If an image contains clutter, only the clutter components are extracted (Section 2.2.1.3) and passed through clutter removal (Section 2.2.1.4) to ensure that non-clutter components are not processed for noise removal, preserving their quality and enhancing efficiency of the approach. As illustrated in Figure 2.4, it is important to determine the best boundary approximation between clutter and content. A bad approximation can result in dependent noise or content deletion. Figure 2.8 shows the phases of clutter removal from an image.

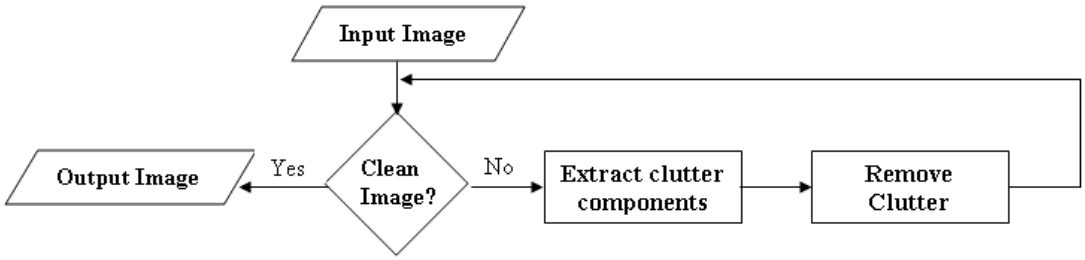


Figure 2.8: Clutter Detection and Removal Flowchart.

2.2.1.1 Background: Distance Transform approach

The proposed algorithm is based on histograms of distance transforms to predict the best approximation to clutter-content boundary. The histograms are based on integer bins, which requires integer approximations to distance transform. Let p be a pixel in the document image I, located at (x, y) position, where $0 \leq x < image\_height$ and $0 \leq y < image\_width$. Let $d(p_i, p_j)$ be a positive definite, symmetric and triangular measure of the distance from pixel p_i to p_j [75] such as the Euclidean distance. Rosenfeld and Pfaltz [75] proved that *octagonal distance*

d_o is a better approximation to Euclidean distance than *city block, square, hexagonal* and *ceil of Euclidean* distance functions. Furthermore, *nearest integer to Euclidean* and *floor of Euclidean* are not distance functions as they violate triangular property. Octagonal distance is defined as:

$$d_o = \max([2(|x_i - x_j| + |y_i - y_j| + 1)/3], \max(|x_i - x_j|, |y_i - y_j|)) \tag{2.1}$$

Distance Transform [19] associates distances to every pixel of a set P from other sets as follows:

$$D_P(p) = \min_{q \epsilon I}(d_o(p, q) + f(q)) \tag{2.2}$$

where initially,

$$f(q) = \begin{cases} \infty & \text{if q } \epsilon \text{ P} \\ 0 & \text{otherwise} \end{cases}$$

For binary images, there are only two sets of pixels, depending on foreground (fg) background (bg) pixels:

$$I = \{P, P'\}, \quad P = \{p | I(p) = fg\}, \quad P' = \{p' | I(p') = bg\}$$

The distance transform is used both for detection and removal. We define D_I as the foreground distance transform of image I, where foreground pixels are labeled by their distance to the closest background boundary and all background pixels are labeled 0. $D_{I'}$ is defined as the background distance transform of image I, where background pixels are labeled by their distance to the closest foreground boundary and all foreground pixels are labeled 0. The distance transform can be computed efficiently with a two pass algorithm presented in [74].

2.2.1.2 Pre-processing

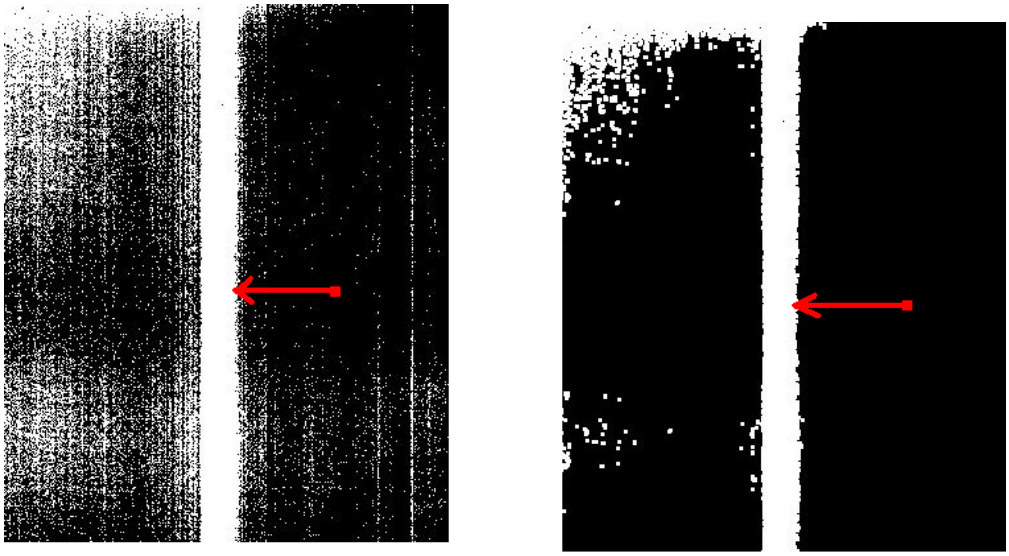
Distance transformations on the foreground pixels labels each pixel with its nearest distance to the background. With the goal of finding the optimal boundary between clutter and attached content, presence of any small opening inside clutter will effect the distance transform map inside it. Unfortunately, clutter does not always form a smooth boundary with the background. This means, as we move out from the center of clutter towards a point on its boundary, instead of a contrasting edge (clean step function in terms of foreground and background), the clutter may have a fragmented appearance (Figure 2.9a). Figure 2.10 shows distance transform values along a path from center of clutter (max) to a point on its boundary (min), in the presence and absence of fragmentation. Distance transform produces an appearance of phantom clutter-edges due to this fragmentation which affects later stages of clutter detection and removal.

Two prominent methods are worth considering but have shortcomings:

1. Median-filtering [28]: The fragmentation is a continuous flip-flop of pixels in background and foreground. Median filtering assumes a much higher frequency of one set of pixels. Hence, this method is not suitable for the problem.

2. Morphological closing: This is challenging due to the two unknowns - the size of the structuring element and number of passes required.

We use *distance transform based closing*. A background distance transform ($D_{I'}$) is first applied on the image. Each opening is labeled with its maximum



(a) (b)

Figure 2.9: (a) shows a clutter with fragmented boundary (b) shows a clutter with smooth boundary with the background.

distance, refered to as the radius of the opening. The most frequent radius is chosen from the histogram and all background pixels with distances less than this radius are converted to foreground producing an image I_c.

Figure 2.9b shows the pre-processed result of clutter in Figure 2.9a. This process also affects and thickens the text in I_c. The marked clutter components from I_c are then overlaid on the original image I, keeping the text in I uneffected, while filling up the clutter openings.

2.2.1.3 Clutter Detection

Clutter, by definition, is larger than the maximum text-stroke width present in the document, whereas thickness of ruled-lines, salt-n-pepper, stray-marks or bleed-through can be of the order of text-stroke width. It is interesting to note that this property of clutter differentiates it from other types of noise and text.

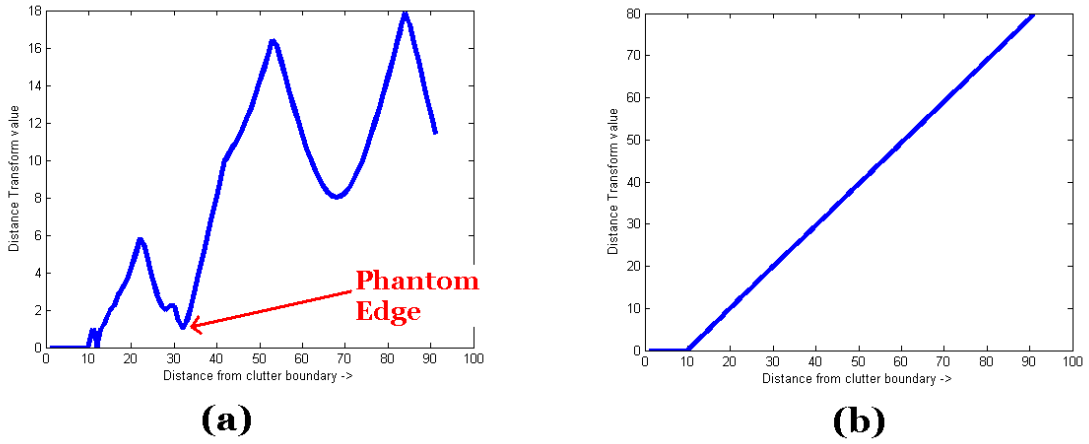


Figure 2.10: Figure shows foreground distance transform values along a path from center of the clutter to its boundary in case of (a) a fragmented boundary (b) smooth boundary.

The clutter detection process is independent of the general shape, and only requires that the size is twice as large as the stroke-width. Foreground distance transform labels each foreground pixel with a distance to the closest background boundary. The foreground pixels, with associated distances less than half of the maximum transform distance in the image, are converted to background. This results in an image called the *residual* (Figure 2.11(b)). It can be computed as follows:

1. Perform distance transform D_{Io} on the original binary image Io, as illustrated in Equation 2.2

2. Calculate the maximum value $dtMax = \max(D_{Io})$

3. Set all pixels p with $D_{Io}(p) < dtMax/2$ to background. The residual image Ih is obtained.

Clean text and cluttered documents can be differentiated on the basis of

Figure 2.11: (a) An image with clutter and text. (b) The distance transform on the image with distances normalized to gray values [0-255] (c) Result of residual process.

what kind of residual image they produce. In a cluttered image, the residual process will remove all text and non-clutter noise pixels from the document and will leave behind only a core of each clutter component. On the other hand, in the absence of any clutter, text strokes will be reduced to half their maximum width, maintaining a text-like pattern (albeit broken). The basic differentiating properties of a clean and clutter image are thus enhanced through this residual process, hence producing better differentiating features. The features of these residual images are calculated using the properties of their connected components shown in Table 2.2. These properties of residual images, *Ih*, are used as features to detect if the original image, *Io*, contains clutter. We train a 2-class SVM on two sets of residual images from clean and clutter documents. Since *number of instances* $\gg$ *number of features*, we use an RBF Kernel [25]. The residual of test image is then classified as having or not having clutter. Figure 2.14 shows the clutter detection and removal model. Once a residual image *Ih* is classified as having clutter, its components are replaced with the corresponding (and larger)

Table 2.2: Differentiating properties of residual images of clean and clutter document.

Features of CC	**Clean Image Residual**	**Clutter Image Residual**
Number	High	Low
Average size	Low	High
Variance in size by average Size	Low	At extremes (zero or high)
Variance in positions of centroids of CC	High	Low
Average ratio of area by perimeter	Low	High
Ratio of CC before and after residual	Close to 1	High

connected components from the original image *Io*. Resulting image *Ic* has only these candidate clutter components from their original image *Io*.

2.2.1.4 Clutter Removal

Our goal is to identify those pixels of the clutter components which belong to the clutter, and isolate the non-clutter (text) pixels if there are any. During clutter removal, we reverse the process to determine the point at which the clutter boundary meets the text like pixels. We observe that if we "regenerate" the clutter according to the distance transform from the residual components obtained in the previous step, by introducing the pixels from the original component for successive distances, then the clutter pixels will be encountered at roughly the same rate in every step, as when we removed them. This is evident when we examine the number of regeneration steps that would be required to encounter

each unique foreground distance contour.

As we approach the text-clutter boundary, this no longer holds true, because as we encounter text contours, the number of steps required for regeneration increases significantly where the text protrudes from the clutter. Alternatively, we can consider for an original removal step how many regeneration steps (unique distances) would be required to regenerate it. The original removal step (or distance contour) at which this number increases sharply is the minimum distance ρ from clutter's boundary at which all text is completely removed and the shape of the boundary is best preserved (Figure 2.12).

This process can be summarized as follows:

1. Compute D_{Ic}
2. Compute $D_{Ih'}$
3. $\forall\ p\ \epsilon\ \{Ih' \cap Ic\}$,

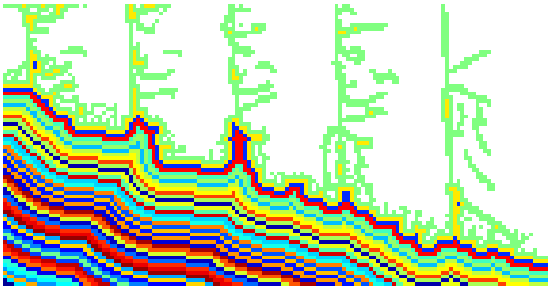
   $d = D_{Ic}(p)$,

   Contour set $C(d) = C(d)\ \cup\ D_{Ih'}(p)$
4. $f(d) = |distinct(C(d))|$

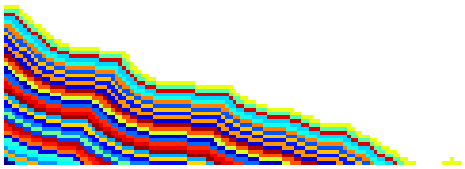
As shown in Figure 2.13, moving outwards towards the boundary of clutter-component, there is a sharp rise in $f(d)$ at ρ. This function is a monotonically decreasing function. $f'(d)$ is the rate of change of the function, which slows down at ρ. If $g(x) = f''(d)$, ρ is the index of *first* maxima of g(x).

$$\frac{d}{dx}(g(x)) = 0, \quad \frac{d^2}{dx^2}(g(x)) > 0 \tag{2.3}$$

(a) Clutter Image

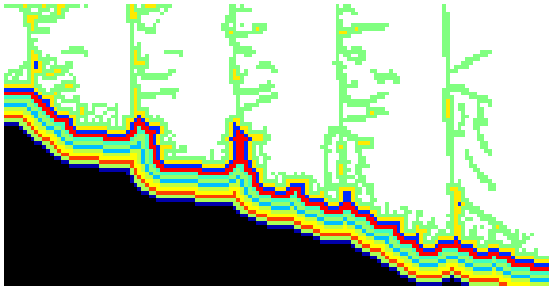


(b) Foreground distance transform. Each colored contour consists of pixels with similar distance from the background (clutter component's boundary)



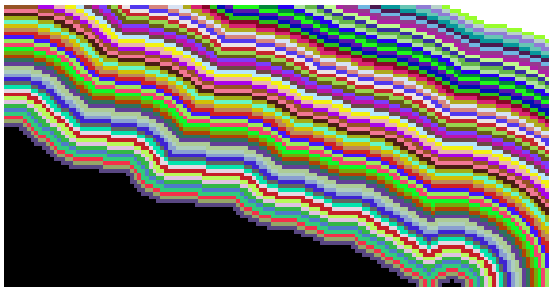
(c) Removing pixels with DT < half of the maximum

(d) Obtaining Residual Image

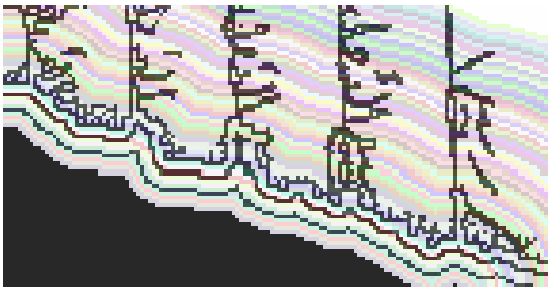


(e) Overlaying residual image on removal steps (contours). The darker contour is the nearest contour to the clutter's irregular boundary which does not intrude in content. This is the best approximation to the clutter-content boundary

(f) Selecting three removal steps (distance contours) at various distances from clutter's boundary

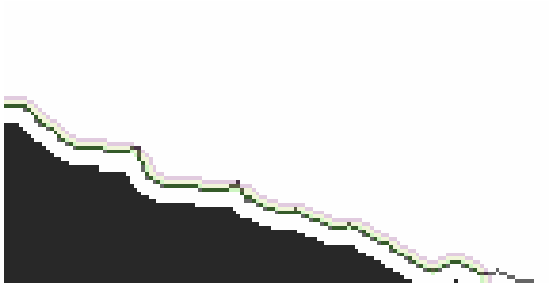


(g) Regeneration steps from residual image



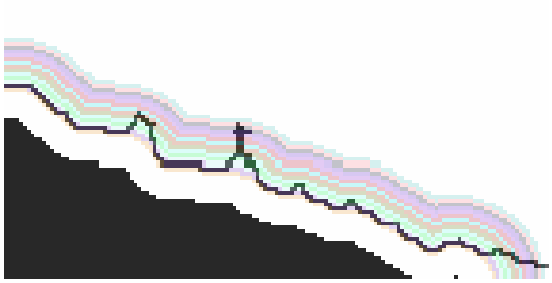
(h) Regeneration steps overlaid on the selected removal steps

Figure 2.12: Clutter Removal Process.

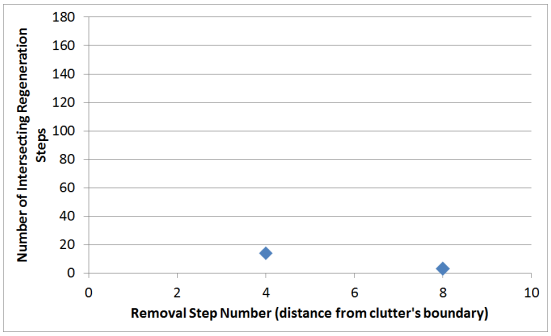


(i) Number of regeneration steps overlapping with 8th removal step = 3

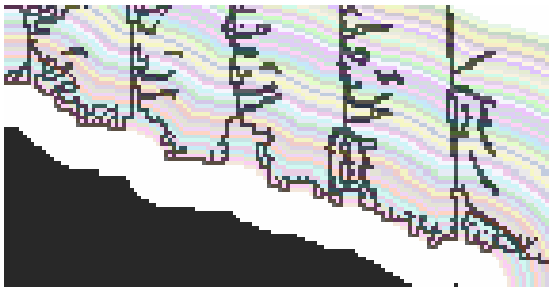
(j)



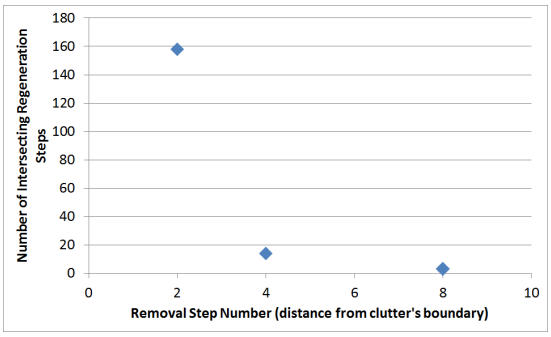
(k) Number of regeneration steps overlapping with 4th removal step = 14



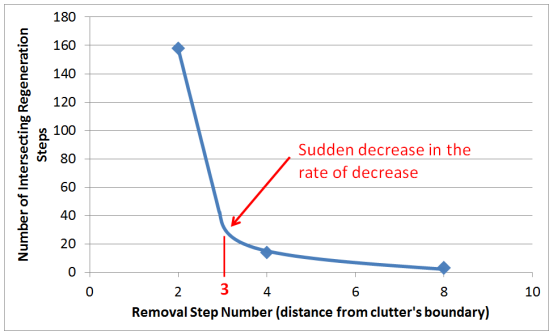
(l)



(m) Number of regeneration steps overlapping with 2nd removal step = 158



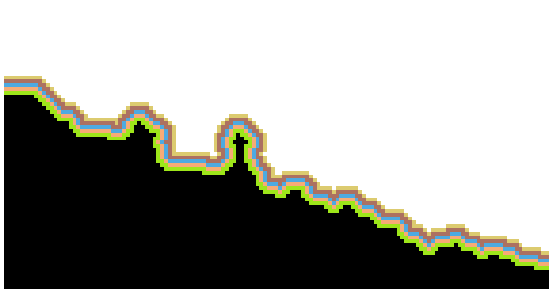
(n) Number of regeneration steps grow exponentially at the removal step separating clutter from the text pixels



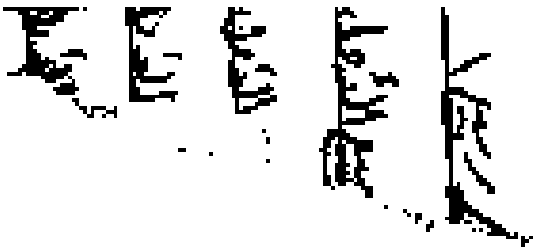
(o)

(p) Best approximation to the clutter-text boundary at 3rd removal step

Figure 2.12: Clutter Removal Process.



(q) Regenerating clutter in 3 steps from the best approximated boundary



(r) Removing the detected clutter from the original image

Figure 2.12: Clutter Removal Process.

It is not important that residual components maintain the exact shape of the clutter. The point of *first* sudden drop in the function can predict the distance from the real boundary. The depth of the drop is proportional to the length of the text-branch. Once this distance ρ is obtained, shrinking and expanding the clutter-component by this distance, identifies the clutter without its text-branches. If ρ is zero, these operations are not performed, as there is no text attached to the clutter and the clutter component can directly be removed.

1. Image Id is obtained by removing all pixels p from Ic such that $D_{Ic}(p) \leq \rho$. Marked Pixels $MP = \{FG(Id)\}$

2. Compute $D_{Id'}$. $MP = MP \cup \{p \, \epsilon \, Id' \, s.t. \, D_{Id'} \leq \rho\}$

3. From Io, set MP to background. Clutter from Io is removed while preserving the text

It is important to note that since identified clutter pixels are eventually removed from original image Io, any affect on text pixels due to pre-processing (Section 2.3.2) is not reflected in the final image. The final image has no text degradation and is clean of clutter which was originally present in the image.

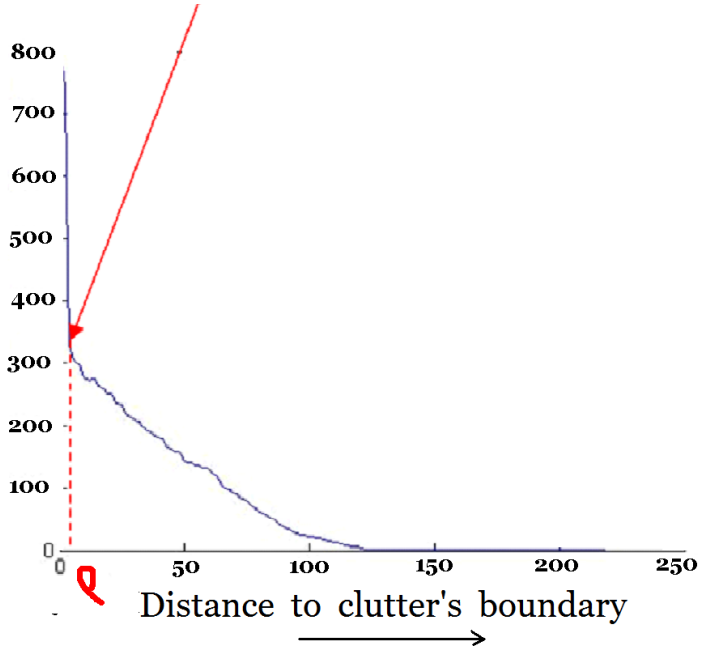


Figure 2.13: Frequency graph showing a sharp rise at ρ.

In a peculiar case, where several clutter components of sizes in multiples of each other occur in the same document, the first residual process may remove the smaller clutter components (along with the text) and may leave only the core of the biggest clutter component in the residual image. The clutter removal process following that, is oblivious to the smaller clutter components and cleans only the biggest clutter component. The resulting clean image is again checked for clutter (through residual process and clutter detection). This process is performed iteratively until a residue of a clean document is detected, as shown in Figure 2.14.

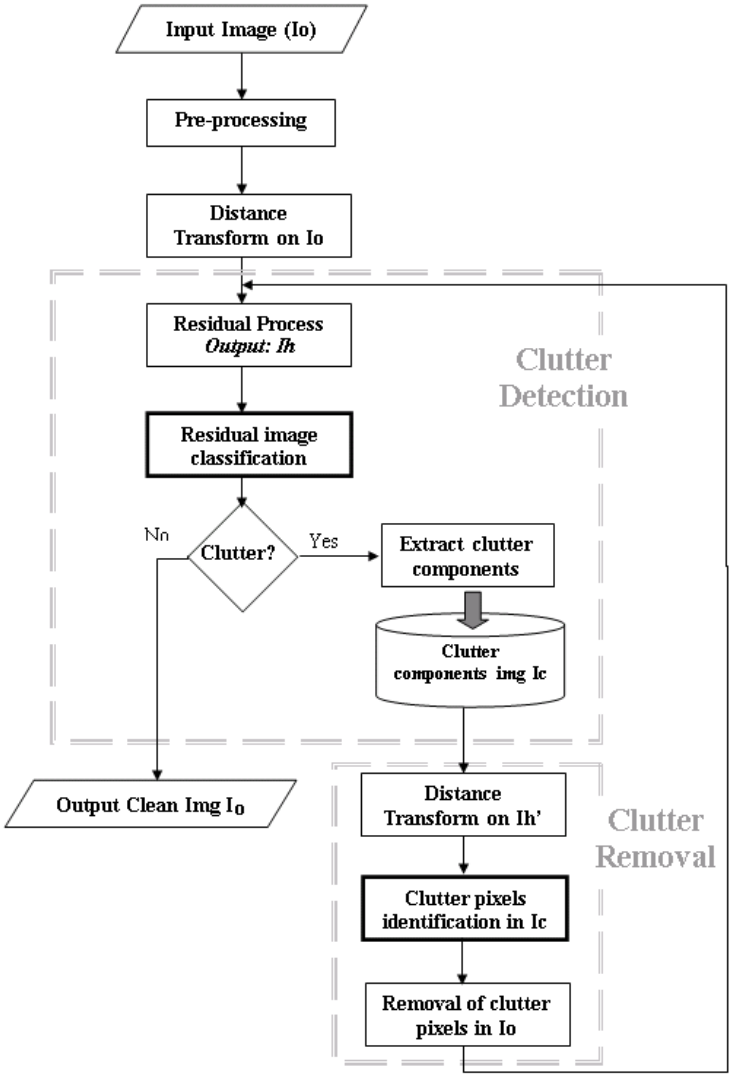


Figure 2.14: Clutter Detection and Removal.

2.2.2 Evaluation

We use two metrics to evaluate our clutter removal approach. The first is a pixel-based metric based on the percentage of clutter pixels removed. The second is purposive, where we evaluate the improvement in successive stages of document processing due to clutter removal.

2.2.2.1 Datasets

We evaluated the clutter detection and removal approach on a dataset of printed and handwritten documents in English and Arabic scripts from four different sources. The various forms of clutter discussed in the Section 2.1 interacts with the text in various ways:

1. No text interaction: Clutter that appears as marginal noise due to scanning of skewed document or because of the gap between the edges of paper and the scanner bed, is around the content and generally does not interact with text. This type of clutter (as shown in Figure 2.1a,d) is easiest to detect and remove without compromising any content.

2. Word-level interaction: Ink seeps, smudges and blobs typically interact with text at the word-level (Figure 2.1c). Detection of this type of clutter is often challenging. Since the text attached is comparable to clutter's size, the precise determination of the clutter boundary is important.

3. Content-level interaction: When clutter appears all over the document due

to bad thresholding or due to document degradation in historic documents, it interacts with a lot of foreground content (as shown in Figure 2.1b,d). If the document has ruled lines (Figure 2.2), a huge loss in content may appear if text attached to clutter is not saved during clutter-elimination.

The datasets used are as follows:

1. Arabic Noisy Handwritten (D_1): This dataset consists of handwritten Arabic text, stamps, logos and figures with noise in the form of stray marks, clutter and salt-n-pepper. The zones are polygonal.

2. English Machine-Print (D_2): The second dataset is from the University of Washington III(UW-III) database [39]. The database consists of 1600 English document images and is widely used in the document analysis community. It contains 10 different zone types - chemical drawing, small text, symbols, drawing, halftone, logo or seal, map, math, table and large text. The selected documents were primarily scanned publications or journals with rectangular zones.

3. Complex English (D_3): The third dataset consists of highly degraded and noisy English documents. The documents consist of forms, handwriting annotated printed text, tabular columns, letters and memos. The zones are primarily polygonal.

Apart from these clutter datasets, we used two clean datasets. Two important reasons to use them were the availability of text-line annotation for purposive evaluation and for training clean documents for clutter detection.

4. Arabic Clean Handwritten (D_4): The dataset was clean with primarily text-lines as the only content.

5. Arabic Clean Mixed (D_5): The dataset consists of textual content, stamps, signatures and noise in the form of rule-lines and speckle.

2.2.2.2 Pixel-based Evaluation

Protocol and Results: In order to evaluate clutter detection separately, we sampled a representative set of 50 images with all forms of clutter and a set of 50 clean images from our dataset. We divide the set of 100 images into a training and testing set in ratio 3:2. 30 images from each set are used for training. Clutter detection pixel-level precision and recall on the remaining 40 images is reported as 92.5% and 100% respectively (Table 2.3).

For clutter removal, we used the LAMP's xml-based GEDI tool [105] for pixel-based labeling and visualization producing ground-truth and result files (containing polygonal zones) in the GEDI XML format specification [53]. GEDI is a public domain ground-truth editor and document interface for scanned text documents. It's interface maintains a one to one correspondence with XML files and the corresponding image files. Different types of zones can be created and visualized using a custom set of attributes. Each image is labeled into clutter and non-clutter (text, ruled-lines etc.) pixels. Since clutter components are much larger than text components, clutter pixels occupy 80.6% of the pixels in the dataset. We calculate precision and recall of our clutter removal algorithm using

the following metrics to evaluate the effective gain in accuracy.

$$Precision_N = \frac{Noise\ Pixels\ Removed}{Total\ Pixels\ Removed} = \frac{TP}{TP + FP}$$
$$Recall_N = \frac{Noise\ Pixels\ Removed}{Total\ Noise\ Pixels} = \frac{TP}{TP + FN}$$

We achieved precision and recall accuracy of 99.41% and 92% respectively for clutter-pixels. The results are depicted in the Figure 2.15 and Table 2.4. The high value of precision highlights the restrictive nature of our approach which does not consume text pixels during removal. The 8% error in recall is primarily due to the severe fragmentation in clutter components, especially near their boundaries.

	Clutter Detection
Recall	100%
Precision	92.5%

Table 2.3: Detection Accuracy.

	Clutter Removal
Recall	92%
Precision	99.41%

Table 2.4: Removal Accuracy.

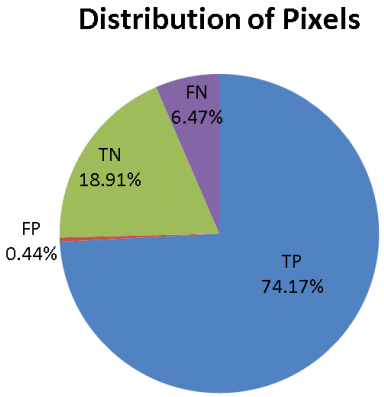


Figure 2.15: Pixel Distribution.

Figure 2.16 shows the clutter removal results of images in Figure 2.1. Figure 2.17 shows the removal of clutter after precise detection of non-linear clutter-text boundary.

Performance: Clutter removal performance was measured across different image resolutions and foreground content for 200 document images from above

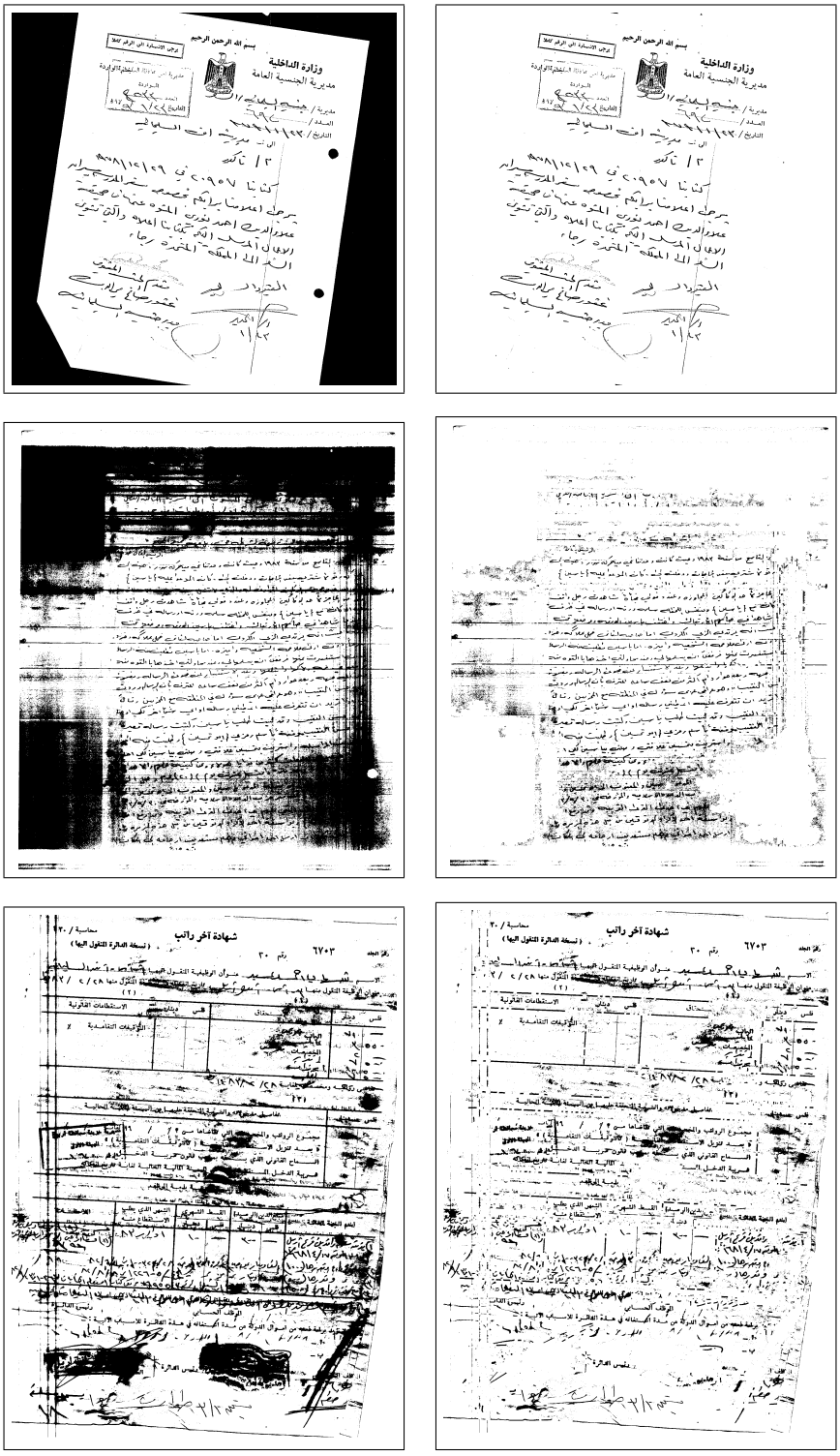


Figure 2.16: Results of our clutter removal approach: The left side shows the cluttered images and the right side shows the corresponding cleaned images as a result of our clutter removal approach.

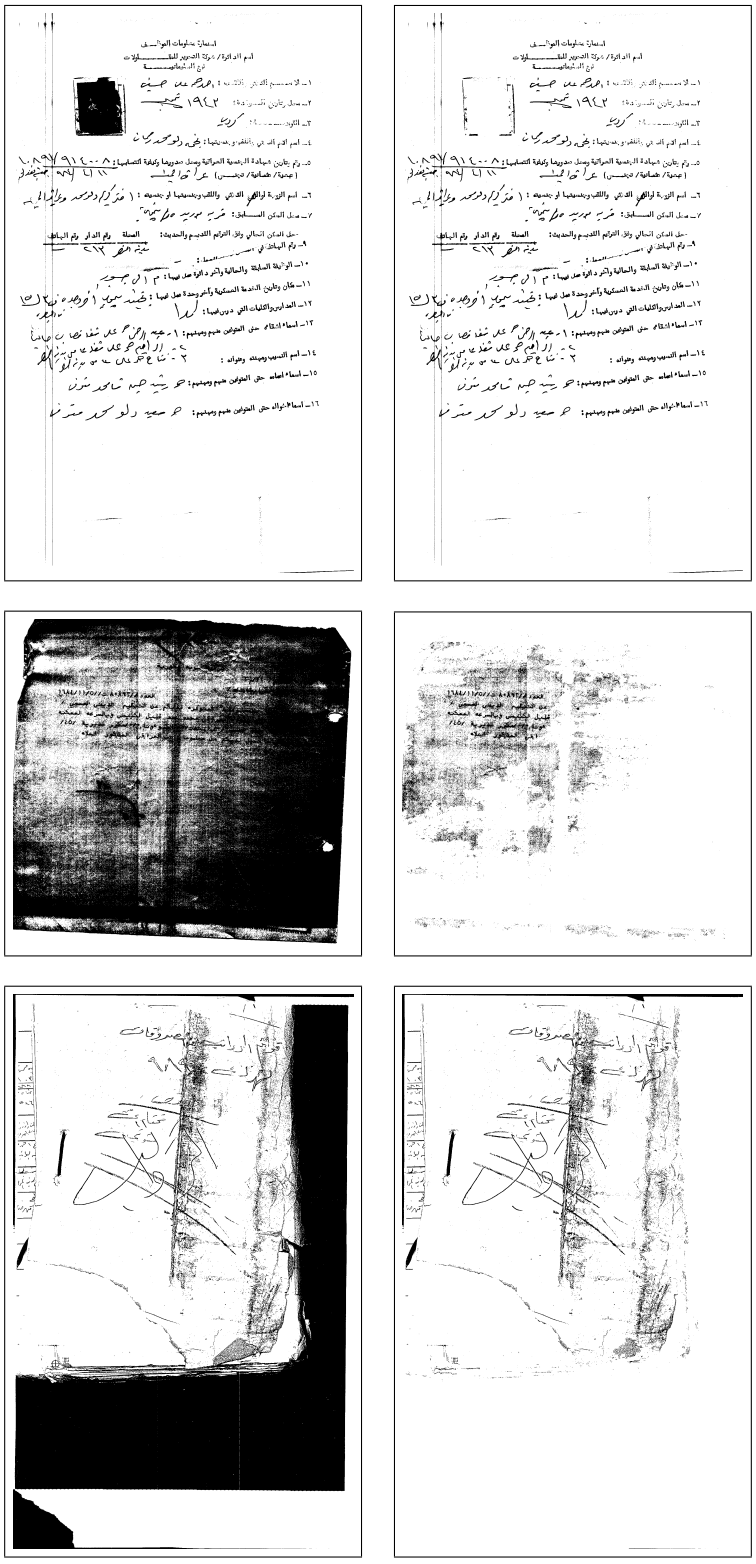
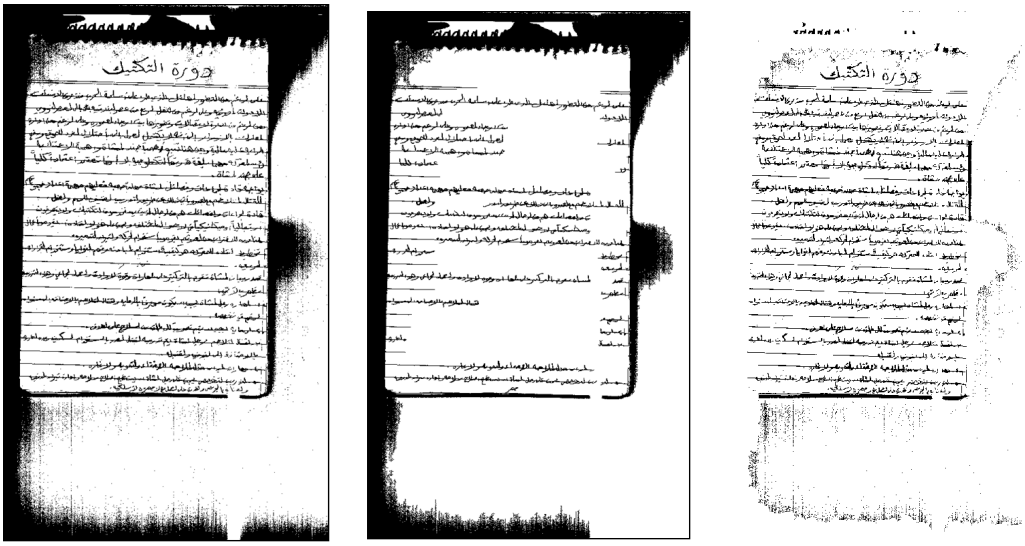


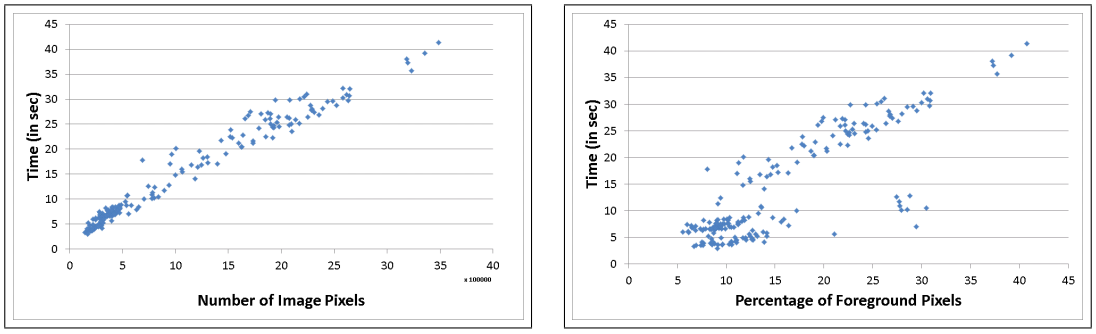
Figure 2.16: Results of our clutter removal approach: The left side shows the cluttered images and the right side shows the cleaned images as a result of our clutter removal approach.



(a) (b) (c)

Figure 2.17: (a) Clutter document image (b) One big connected component containing clutter and majority of content (c) Clutter removed with text preserved.

datasets. We verified that the performance varies linearly with these two parameters and does not grow exponentially for better quality or dense content images (Figure 2.18). On average clutter removal algorithms take 14 seconds on an Intel 3.00GHz (single core), 1GB RAM system under normal usage conditions.



(a) (b)

Figure 2.18: Performance varies linearly with image size and foreground content.

2.2.2.3 Purposive Evaluation

A second evaluation measures the impact of clutter removal in successive stages of the document processing chain. In document processing, the next step after noise removal is often text extraction, followed by word segmentation and finally character recognition. Clearly, text-extraction or line segmentation is the stepping stone to later stages. Any error in this stage will effect successive ones. In order to evaluate the effect of clutter noise in documents, we compared in-house line segmentation algorithms on clean and clutter documents.

We used a component-based handwritten Arabic text-line segmentation algorithm using Affinity propagation [48]. This is a graph-based method for extracting handwritten text lines in monochromatic Arabic document images.

Adding clutter: We randomly sampled 250 images of cleaner handwritten Arabic data containing mainly textual content (D_4) and 200 images of noisy handwritten Arabic data containing textual content, stamps, signatures and noise in the form of rule-lines and speckle (D_5).

Clutter is added to all 450 images in two forms:

1. Border clutter

2. Random number of clutter blobs of random sizes at random places interacting with text

Clutter detection and removal is performed on these images to produce a *clean* dataset.

Protocols and Results: On cleaned documents, we noted an improvement of 8% in the case of the pure-handwriting dataset (D_4), while an improvement of approximately 2% in inherently noisy mixed-dataset (D_5). This also underlines the sensitivity of the line segmentation algorithm on other forms of noise (ruled-line, speckle) as well.

Table 2.5: Purposive evaluation of clutter on line segmentation algorithm.

	D_4		D_5	
	Noisy	Clean	Noisy	Clean
Precision	82.00%	90.36%	46.36%	49.58%
Recall	79.58%	86.32%	39.32%	41.66%

2.2.2.4 Error Analysis

The clutter removal process cleans up majority of the clutter from a document image. However, there are primarily two sources of errors, which effect clutter's recall (precision is very high):

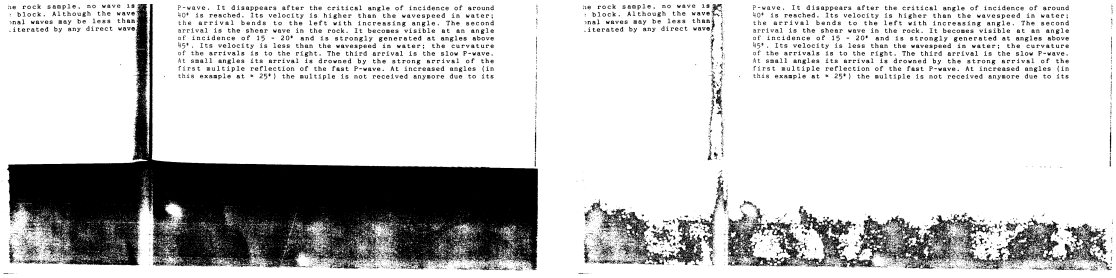
Stroke-like noise protrusions: The restrictive nature of clutter removal algorithm treats any stroke-like protrusions, with width similar to that of text, as text and errs by preserving them during removal (Figure 2.20). In order to avoid such residual noise, one possible solution is to extract features of the detected text-pixels, based on which a decision to delete or retain them can be made.

Figure 2.19: Line segmentation results on clutter and cleaned images.

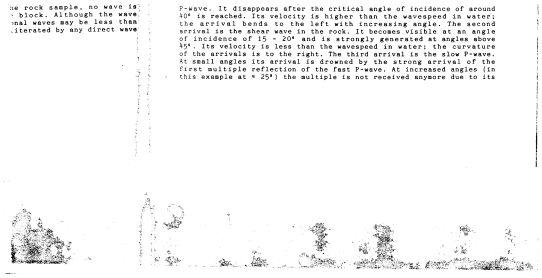
(a) (b)

Figure 2.20: (a) Text-line image of spiral bindings at the top of the document image are treated as text (b) Clutter removal fails at removing text-line protrusions.

Fragmentation: Fragmented boundaries of clutter produces an appearance of phantom clutter-edges as discussed in Section 2.2.1.2. While our pre-processing step creates a contrasting edge for majority of cases, it fails to address highly variant fragmentation mixed with Stroke-like Pattern Noise (Figure 2.21).



(a) Clutter with severly fragmented boundary (b) Clutter removed without pre-processing step



(c) Clutter removed after pre-processing

Figure 2.21: Stroke-like Pattern Noise is left behind after complete clutter removal process.

2.3 Stroke-like Pattern Noise

2.3.1 Prominent Text Components

Intuitively, text has the following distinguishing characteristics: 1) text possesses certain frequency and orientation information; 2) text shows *spatial cohesion* - a set of strokes appear together to form words or phrases [97]. At component-level, many of these stroke components, in cohesion, contain prominent textual features like length, critical points, cusps, arcs and curves. Such text components with independent features are called prominent text components (PTC). PTCs can be identified as text components individually and do not require any neighboring context. However, many smaller components, like diacritics, use their positions with respect to PTCs and stroke-widths to identify themselves as textual content. These two properties of the smaller components are tightly coupled with the prominent textual components (Figure 2.22).

Figure 2.22: Red (dark gray) and black components depict PTCs and non-PTCs respectively.

2.3.2 Noise Removal using a Content Understanding Technique

We use the above listed text properties to devise a two-phased component-based divide-and-conquer approach to extract text components from a noisy binary document image using a minimal set of training samples. In the first phase, we classify prominent text components (PTCs) using a supervised classification approach. Aiming at the script-independent features of text strokes, a generalized feature set is devised to classify the PTCs using a limited training dataset. Later, based on the stroke-width and cohesiveness properties of these components, smaller text components are filtered out from the noise components using unsupervised k-means clustering.

2.3.2.1 Supervised Prominent Text Component Classification

Prominent text components exhibit script-independent and context-independent properties to distinguish themselves from other types of content in a binary image. Apart from area, perimeter, convex-area of each component, orientation of the fitted ellipse, it's major and minor axis lengths and eccentricity, four more feature descriptors are defined as follows in order to measure the independent shape properties [6].

1. FilledArea: Number of foreground pixels in the bounding box of the component with all holes filled in

2. Extent: Ratio of the pixels in the component to the pixels in the bounding box

3. Solidity: Ratio of the pixels in the smallest convex polygon that are also in the component (=$Area/ConvexArea$)

4. EquivDiameter: Diameter of the circle with the same area as the region (=$sqrt(4 * Area/pi)$)

These features are normalized by the average size of the connected components and scaled to the range $[0, 1]$. The components are labeled as PTCs and non-PTCs (includes smaller text components and noise) on a limited set of training samples, and sent to the feature extraction module. LibSvm library [25], is then used to classify the two set of classes. A selective number of features used over a large number of components ($|features| \ll |instances|$) implied using an RBF Kernel for classification in order to nonlinearly map data to a higher dimensional space.

After classification, the results are sent to the second-phase to selectively remove noisy components from the image.

2.3.2.2 Unsupervised Small-Component Classification

In order to filter small text components from a pool of non-PTCs, we compute two characteristics of all components - their stroke-width and cohesiveness with respect to PTCs. These are computed efficiently using a distance transform approach [10]. The distance transform labels each pixel of the image with the distance to the nearest pixel of different gray-value. For a binary image, foreground distance transform, D_1, labels each pixel with its nearest distance to the background pixel, thus producing a distance map with increasing distances from

the edge of each component to it's center. Similarly, $D_{I'}$ is defined as the background distance transform of image I, where background pixels are labeled by their distance to the closest foreground boundary and all foreground pixels are labeled 0. The distance transform can be computed efficiently with a two pass algorithm presented in [74].

1. Stroke-width: In order to compute this efficiently, we perform a foreground distance transform. Maximum distance value associated with each connected component (CC) defines its stroke-width (sw_{CC}).

$$sw_{CC} = \max(D_I(p)),\ \forall\ p\ \in\ \{CC\} \tag{2.4}$$

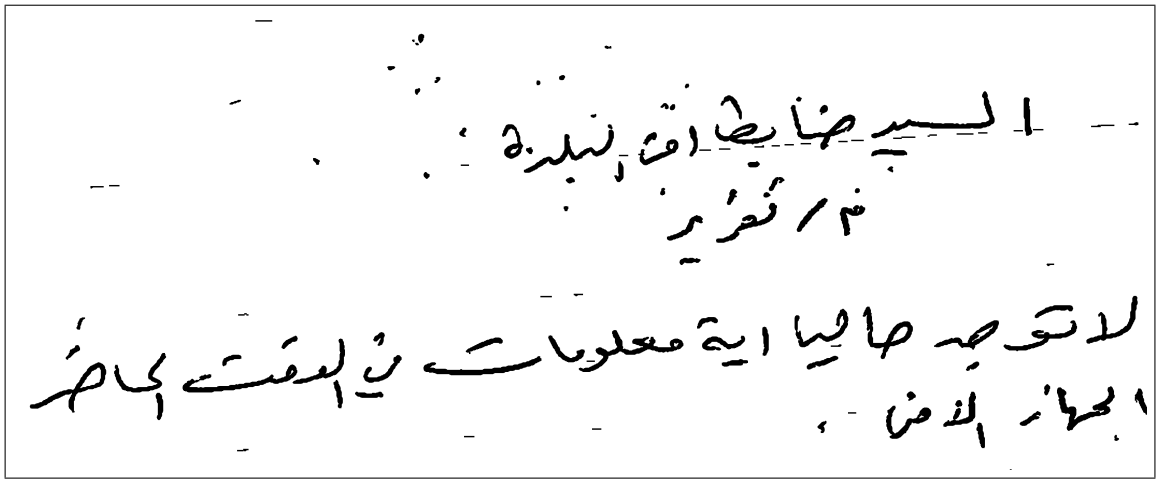
   Mode (highest frequency) of stroke-widths for Prominent Text Components (PTCs) gives the average stroke-width of the document sw_{avg}.

2. Cohesiveness: First, an image with only PTCs is created (I_{PTC}). Performing a background distance transform on that image ($D_{I'_{PTC}}$) assigns each background pixel a minimum distance to the nearest PTC. Cohesiveness (co_{CC}) for each non-PTC is then defined as the minimum distance value associated with the underlaid background pixels.

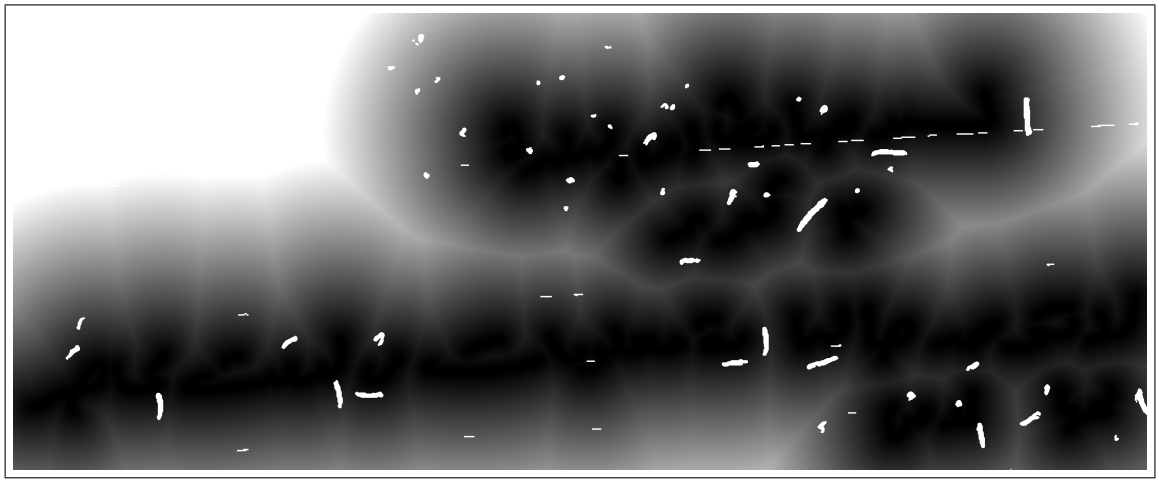
$$co_{CC} = \min(D_{I'_{PTC}}(p)),\ \forall\ p\ \in\ \{CC\} \tag{2.5}$$

   Average distance between each nearest pair of PTC (co_{avg}) is calculated using a distance adjacency matrix.

Figure 2.23b shows the classified non-PTCs overlaid the distance transform map of PTCs for our test image in Figure 2.23a. K-means clustering (k = 2) is



(a)



(b)

Figure 2.23: (a) Stroke-like Pattern Noise, resembling diacritics, present around text components. (b) Image shows classified non-PTCs (smaller text and noise components) overlaid the distance transform map of PTCs. Components nearer the darker regions are closer to the PTCs and vice versa.

applied to non-PTCs based on the defined features ($|features| = 2$). A further verification step is performed with the following rule:

$$if\ sw_{CC} >= sw_{avg}\ \&\ co_{CC} <= cc_{avg},$$

$$classify\ CC\ as\ text - component$$

The small text components are filtered out from non-PTCs leaving the noise components behind. The final result after the second phase is shown in Figure 2.24.

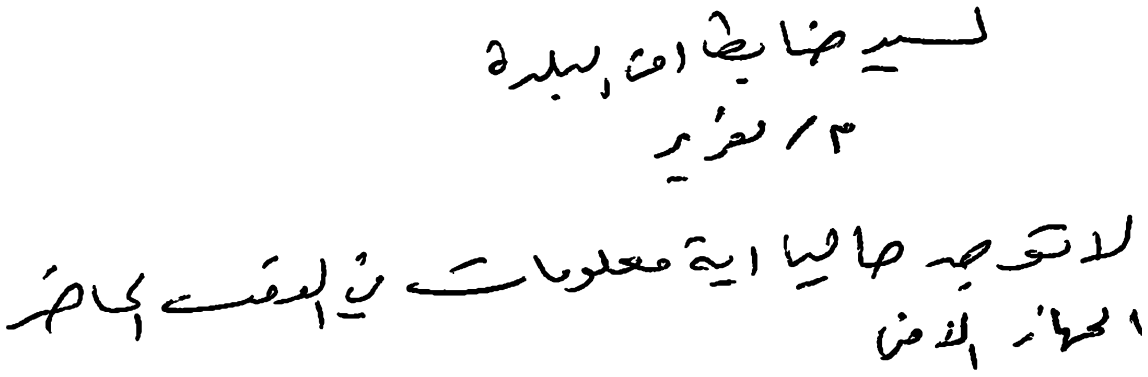


Figure 2.24: SPN removal result of the test image in Figure 2.6. The noise components are successfully removed.

2.3.3 Evaluation

2.3.3.1 Datasets

The dataset consists of printed and handwritten Arabic binary documents. Manual ground-truthing being a laborious job at the pixel-level, we use a representative set of 50 document images containing Stroke-like Pattern Noise (SPN)

from the four sources described in Figure 2.7. Only 2 document images are used to train the SVM for PTC classification to validate our minimal training requirement.

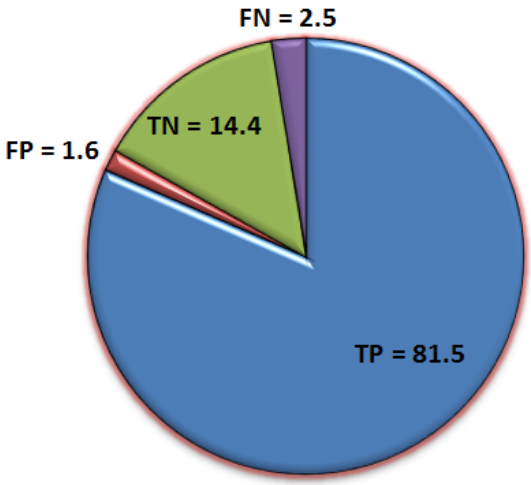
2.3.3.2 Metrics

Pixel-based evaluations are performed in order to assess the accuracy of our approach. SPN being of the size similar to that of smaller text components, and PTCs being much bigger in size, SPN occupies 16% of the pixels in the dataset. We calculate precision and recall of our SPN removal algorithm using the following metrics to evaluate the effective gain in accuracy.

$$Precision_N = \frac{Noise\ Pixels\ Removed}{Total\ Pixels\ Removed} = \frac{TN}{TN + FN}$$
$$Recall_N = \frac{Noise\ Pixels\ Removed}{Total\ Noise\ Pixels} = \frac{TN}{TN + FP}$$

2.3.3.3 Results

We achieved precision and recall accuracy of 86% and 90% respectively for noise-pixels (Table 2.25b). In this component-based noise removal approach, even a few misclassified text-components tend to increase the pixel-level precision error rate due to their comparitively larger component sizes than noisy ones. Using the pixel distribution in Figure 2.25a, we also report the precision and recall accuracy for the remaining text-pixels after noise removal as 98% and 97% respectively. The results of sample documents of each type are shown in Figure 2.26.



(a)

	SPN Removal
Precision	86%
Recall	90%

(b)

Figure 2.25: Results of Stroke-like Pattern Noise Removal (a) Pixel Distribution (b) Accuracy.

Figure 2.26: Results of our SPN removal algorithm: The left side shows the noisy images and the right side shows the cleaned images as a result of our SPN removal approach.

Chapter 3

Page Segmentation and Zone Classification

3.1 Background and State-of-the-art

Current page segmentation algorithms lack the ability to dynamically adapt to local variations in the size, orientation and distance of components within a page. In handwritten and mixed content documents, the size of components vary drastically due to the cursive nature of writing, leading to over or under-segmentation. The goal of this chapter is to highlight the shortcomings of known page segmentation algorithms and describe an integrated approach to the segmentation of these complex pages.

The process of identifying the structure of a document image is called layout analysis, and can be physical (process of dividing the document into physical homogeneous zones) or logical (process of assigning logical roles and relations to detected zones). Page segmentation algorithms fall into the category of physical layout analysis [81] and segment a document page into homogeneous zones, each consisting of only one physical layout structure such as text, graphics, equations, logos or stamps.

Physical layout analysis can be based on pixel or texture segmentation [42], but the general goal is that the final result is a region segmentation. In texture-based segmentation, isolated points or small areas can be classified without regard

to the *connectivity* aspect of an object. In contrast, the work assumes non overlapping geometric zones where document components are separated by white space. Such connected component based approaches use macro level content information, and can be further classified into Manhattan [60, 96, 16, 20], where region boundaries can be laid out on a grid, and more arbitrary non-Manhattan layouts [64, 46, 43].

1. **Manhattan layout:** There are four representative algorithms for Manhattan layout based page segmentation. The run-length smearing algorithm (RLSA) presented by Wong et. al [96] is one of the earliest techniques to segment the page into homogeneous regions. It "smears" within zone components into each other using the perceived text direction and some thresholds, forming a single distinct larger component per zone. X-Y Cut Page Segmentation [60] is a tree-based top-down algorithm, which starts with the entire page as a root. Based on alternating horizontal and vertical projection profiles of foreground pixels, each node is split recursively at the largest valley in the profile until minimum formed regions are contained in its leaves. In 2002, Baird devised a top-down Whitespace Analysis method [16], based on the analysis of the background structure in document images. It generates a sorted list of maximal elongated white-space rectangles which are unified until a stop rule is satisfied, to generate a sequence of enclosed segmentations. Constrained text-line detection by Breuel [20] builds on white-space and X-Y cut, with an added consideration of column-

separations (gutters) while grouping text-lines. This has a direct advantage over smearing technique where presence of a known gutter may skew the thresholds. For documents with known structure, such as books, these techniques can work very well.

2. **Non-Manhattan layout:** The focus of this chapter is on more generic non-Manhattan layouts that are prominent in handwritten or mixed content documents. Connected component analysis, skew [35, 64, 40] and analysis of background [69, 14, 63] have been used by researchers to perform page segmentation on non-Manhattan layouts. Of these, O'Gorman [64] and Kise et. al. [46] are the most widely cited algorithms for geometric page segmentation on non overlapping zones. O'Gorman's Docstrum algorithm performs transitive closure on within-line connected components to obtain lines and then on lines to form regions. The thresholds for transitivity are based on the properties of distance and angle of each connected component with it's K nearest neighbors [64]. Kise's algorithm based on Voronoi regions is the first algorithm to use properties of components (in terms of area) in addition to their white-space separations. Details in these algorithms highlight their advantages over previous approaches. Evaluation experiments [81] have shown that the Voronoi based approach excels on a mixed dataset of both handwritten and machine printed documents for diverse scripts such as English and Arabic.

Algorithm	Separation: Distance	Separation: Direction	Content: Area Ratio
RLSA	✔	✔	
X-Y Cuts	✔		
Whitespace Analysis Method	✔	✔	
Constrained Text-line Detection	✔	✔	
Docstrum	✔	✔	
Voronoi	✔		✔

Figure 3.1: Comparing various algorithms based on the component features they use for segmentation.

3.1.1 Area Voronoi Based Segmentation

3.1.1.1 Algorithm

The central idea of the original Voronoi algorithm is creation of Voronoi edges between pairs of connected components using an area based Voronoi tessellation [46]. Each edge of the tessellation bisects two points on the contours of different components. A physical zone is a fusion of these Voronoi cells, formed by the elimination of Voronoi edges based on two features:

1. Minimum distance

$$d(E) = \min_{1 \leq i \leq m} d(p_i, q_i) \tag{3.1}$$

where p_i and q_i are pair of points on connected components (CCs) P and Q, constituting i^{th} edge between them

2. Area Ratio

$$a_r(E) = \frac{max\ of\ areas\ of\ 2\ CCs}{min\ of\ areas\ of\ 2\ CCs} \tag{3.2}$$

An edge is deleted if it satisfies one of the following two criteria:

$$\frac{d(E)}{T_{d1}} < 1 \quad or \tag{3.3}$$

$$\frac{d(E)}{T_{d2}} + \frac{a_r(E)}{T_a} < 1 \tag{3.4}$$

where $T_{d1} < T_{d2}$, T_{d1} relates to inter-character spacing, T_{d2} relates to inter-word/line spacing and T_a is the area ratio threshold. The set of remaining edges segment the document image into zones separated by a function of area-ratio and the distance between the components as shown in Figure 3.2 [46].

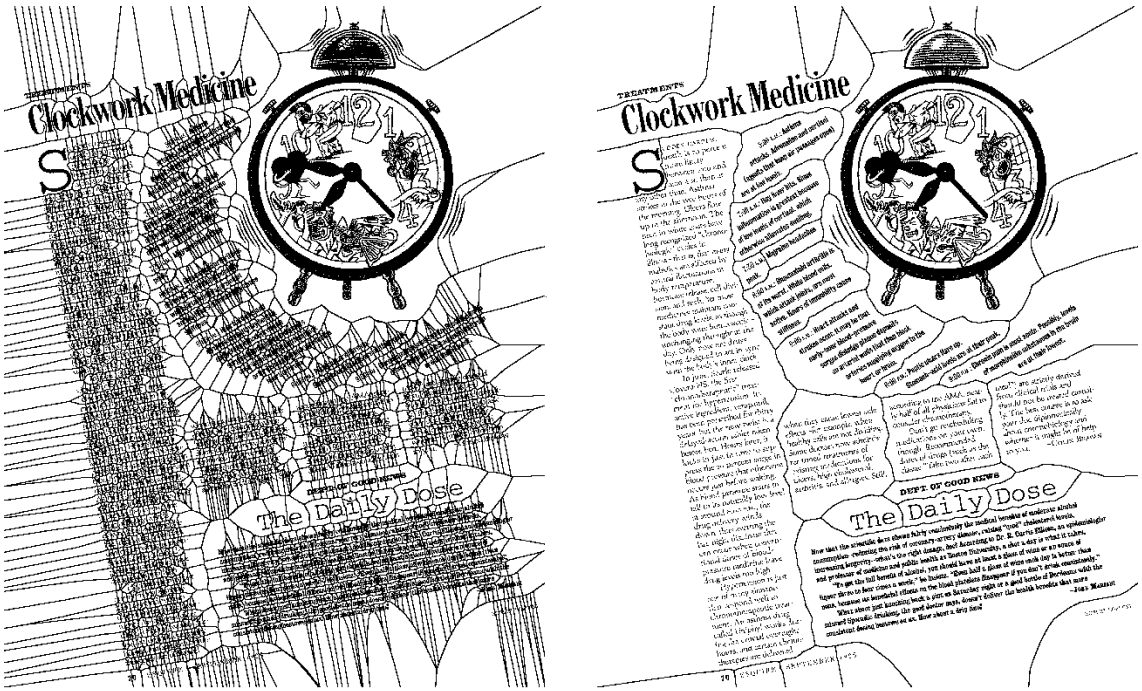


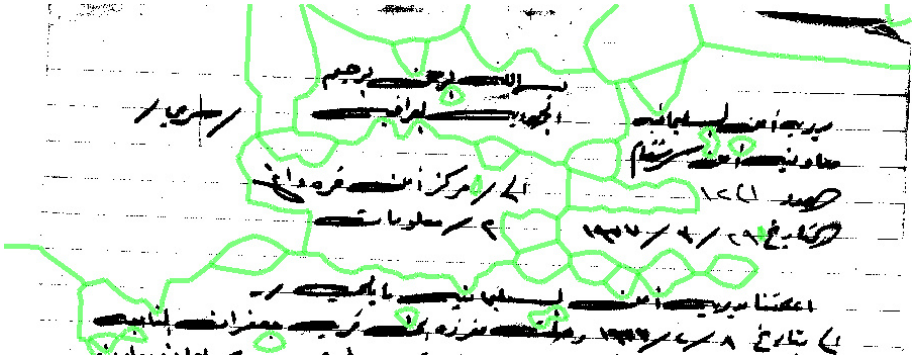
Figure 3.2: (a) Document image overlayed with Voronoi region of each component (b) Weak edges removed based on listed criteria, result is region segmentation [46].

3.1.1.2 Tuning parameters

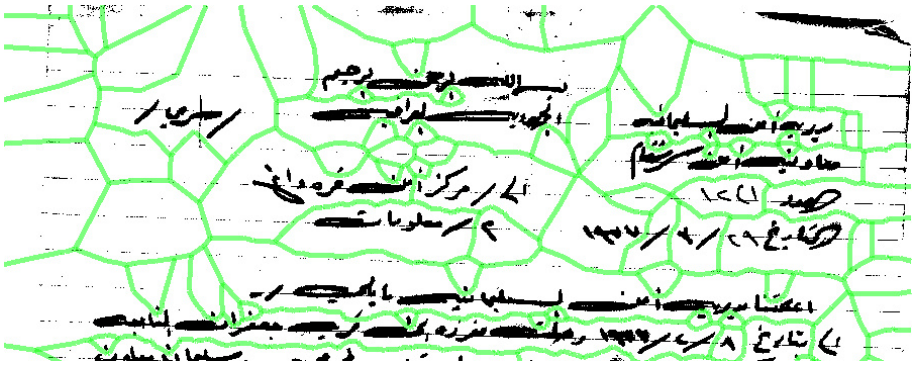
Voronoi based segmentation requires two primary parameters as input. Though some default values can be used, they do not always work for all resolutions or content styles, and thus need to be adaptive.

1. Noise Threshold: For noisy documents with salt-and-pepper noise, edges between smaller noise components and larger text components may persist due to the high area-ratio as shown in Figure 3.8b. Similarly, edges may persist between diacritics, punctuations and corresponding text components. In order to avoid this, the user may input a higher noise threshold such that noisy components or punctuations do not participate in edge formation, preventing over-segmentation. However, choosing this threshold is difficult. Choosing a higher threshold than noisy components may skip smaller or broken text components, thus creating a virtual gap or space at those positions. This adds to the word or line spacing and extra edges may appear within text-regions, leading to over-segmentation. Figure 3.3 shows Voronoi Segmentation of a noisy Arabic handwritten document based on 4 noise thresholds. At threshold 5, even the smallest of noise components participate in the edge formation, hence avoiding any clear region formation. As we increase the threshold, the regions get well separated but again tend to over-segment as higher threshold values start creating virtual gaps within text-regions. The default value is 20.

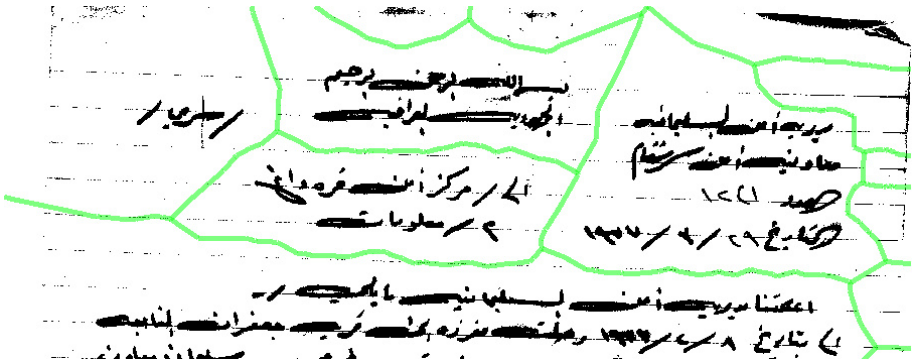
2. T_a: The higher the value, the less it discriminates components of different



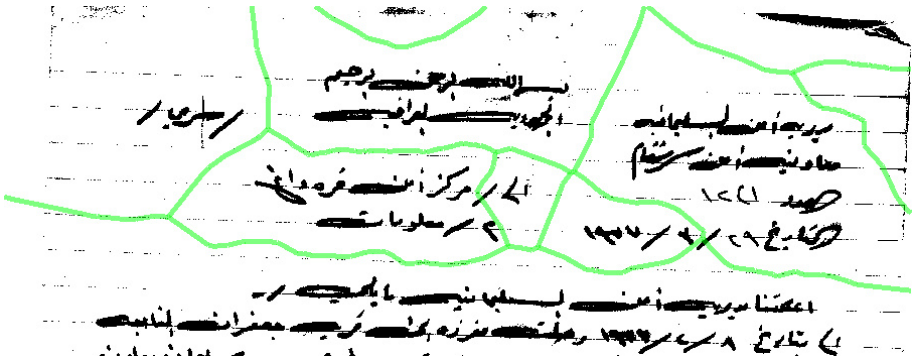
(a) Noise Threshold = 5



(b) Noise Threshold = 20



(c) Noise Threshold = 50



(d) Noise Threshold = 150

Figure 3.3: Figures show sensitivity of Voronoi based segmentation at various noise thresholds (a) 5 (b) 20 (c) 50 (d) 150.

sizes. The default value is 40.

3.1.2 Docstrum Segmentation

Docstrum (or Document Spectrum) is a representation of a document page that describes the global structural features of the page that are used for layout analysis. It is based on bottom-up, k-nearest neighbor clustering of connected components. Each nearest-neighbor pair i, j is described by a 2-tuple $D_{ij}(d, \phi)$ of the distance d and the angle ϕ between the centroids of the two components. It performs transitive closure on within-line components to obtains lines and then on lines to form regions. The thresholds for transitivity are based on these properties of distance and angle of each connected component with its K nearest neighbors [64]. Figure 3.4 [64] illustrates the concept.

The advantage of Docstrum over the Voronoi based approach is its 'semi-local' behavior. This gives the approach a fair amount of independence from skew, different text spacings and an ability to process local regions of different text orientations within the same image. Each component looks at K nearest neighbors to make a decision of its association, unlike Voronoi where decision is solely nearest neighbor based, averaged out globally. In spite of this, Docstrum has been designed mainly for text-only documents and doesn't use any property of the content other than their separation properties, as shown in Figure 3.1.

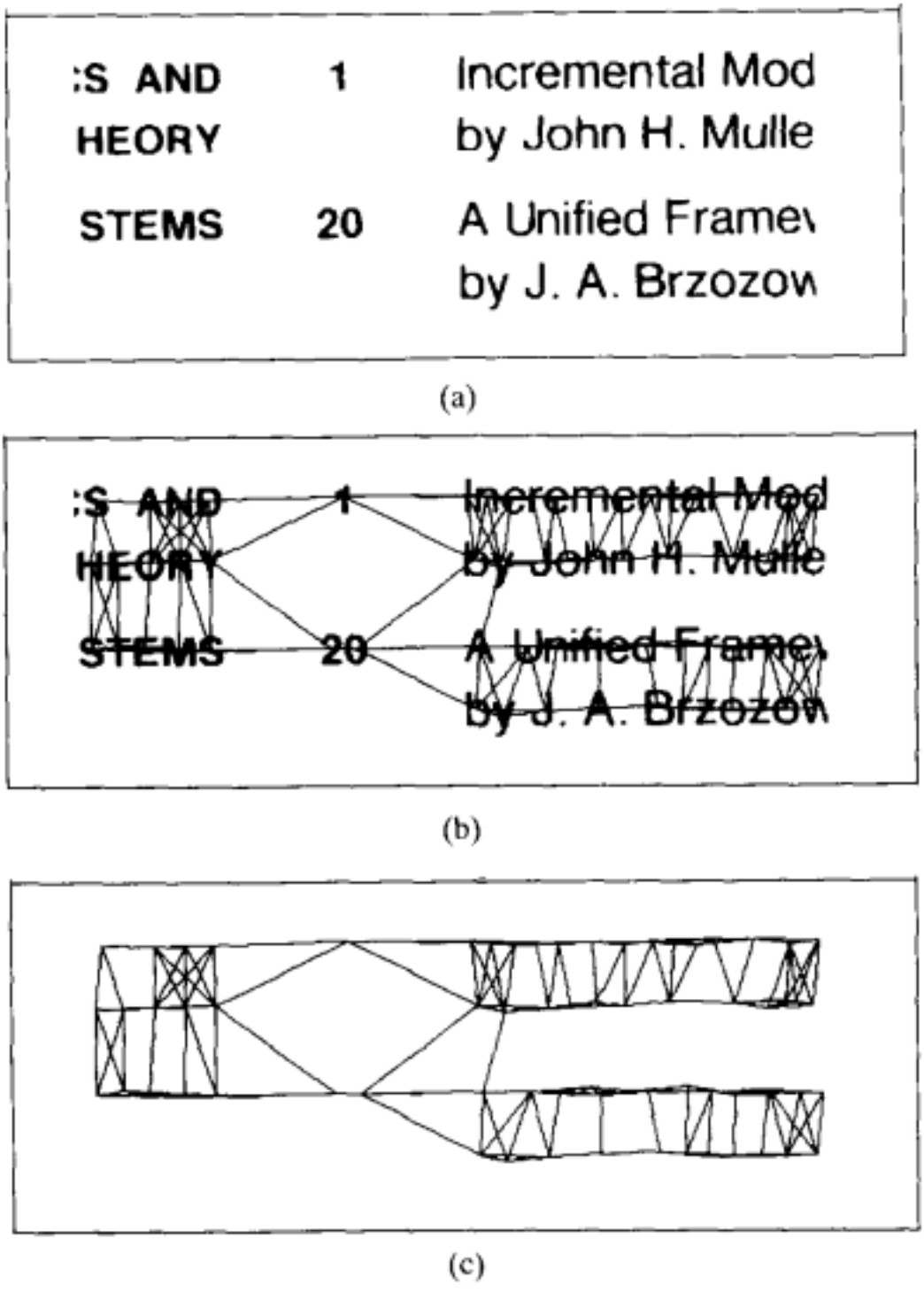


Figure 3.4: Docstrum Segmentation [64], ©1993 IEEE (a) Part of original image (b) Nearest neighbor vectors overlaid on the image (c) Nearest neighbor vectors are shown.

3.1.3 State-of-the-art Limitations

Voronoi-based or Docstrum approaches segment a page based on low-level features like inter-component distance, angle and size-ratios. Whenever an edge which separates two components is removed, the respective zones are fused. This bottom-up page segmentation approach works well when the neighborhood is small. As the regions grow in size, the consequence of merging two zones based on only a single edge feature may change the segmentation significantly. Also, the probability of making a wrong decision increases as the number of edges separating two regions increase with the components in the regions.

In the process of minimizing intra-class difference (content) and maximizing inter-class separation (spatial), the challenge is to come up with an unsupervised segmentation theory of grouping neighboring components based on (a) separation/spatial properties, (b) content and to make context-aware decisions based on these properties. There are three main questions that need to be addressed in designing such a context-aware system:

1. How do we define the local context?

2. How should zone-separation be utilized in accruing context?

3. How should zone-content be utilized in accruing context?

Figure 3.5 adds two columns for local pattern and context properties for a context-aware segmentation system to our previous Figure 3.1 and compares that with all the algorithms discussed so far.

Algorithm	Separation: Distance	Separation: Direction	Content: Area Ratio	Content: Local Pattern	Context
RLSA	✔	✔			
X-Y Cuts	✔				
Whitespace Analysis Method	✔	✔			
Constrained Text-line Detection	✔	✔			
Docstrum	✔	✔			
Voronoi	✔		✔		

Figure 3.5: Comparing state-of-the-art algorithms and their limitations for a context-aware feature-space.

Once the regions (or zones) are determined, it is important to identify a region's content, based on which specialized processing can take place for each zone-type. The primary objective is to extract printed text from other zone-types, for character recognition. In order to achieve this, a novel algorithm for identifying document zone content has been proposed. Low level feature vectors are first extracted from document zones. Partial Least Squares (PLS) [95] is then used to reduce the dimensionality of the feature space and find discriminating features. Rather than using the classic one-against-all or one-against-one approach for zone classification, a new hybrid approach seeking to improve the classification accuracy has been proposed. Support Vector Machines (SVM) are used as the underlying binary classifier.

This chapter is organized into two sections of Page Segmentation and Zone Classification. Section 3.2.1 gives an overview of Voronoi++ model and its context-aware adaptive approach. Section 3.2.2 describes the *hypothesis phase*

(dynamically adaptive Voronoi) and Section 3.2.3 details the *validation phase* (semi-supervised clustering). This is followed by segmentation-evaluation in Section 3.2.4. Section 3.3.1 describes our features for zone classification and the proposed classification method is detailed in Section 3.3.2. This is followed by classification-evaluation in Section 3.3.3.

3.2 Page Segmentation

3.2.1 Voronoi++ Approach

Our model for document image page segmentation uses inter-component separation and local pattern features to form zones. These features are accumulated as zones merge together in order to encompass the newer context. The approach is composed of two phases (Figure 3.6), a *hypothesis* phase where plausible regions are proposed using low-level features and a *validation* phase which verifies the hypothesis using zones' high-level features to make adjustments. Each hypothesized zone can be further split, merged with it's neighbors or left as-is in the validation phase. We reduce this problem to binary classification by eliminating the possibility of a further zone-split in the validation phase.

This requires the *hypothesis* phase to produce an over-segmentation of zones which are unlikely to be split further. The objective of the *hypothesis* phase is to ensure that components belonging to different zones are separated by edges irrespective of false alarms. Edges are forced between any two components which may belong to different zones. These edges are formed based on low-level

features like component separations and variations. This results in an over-segmented image where two neighboring zones (z_i and z_j) are separated by a set of Voronoi edges $e_{i,j}$. This image is represented as a connected graph $G\{E, V\}$, where every zone is represented as a vertex V and any two neighboring zones are joined by an edge-set E of Voronoi edges $e_{i,j}$. The edge-set E is associated with the individual confidences of its edges, returned by the *hypothesis* phase. During the *validation* phase, each vertex is considered with its neighboring vertices for a possible merger, based on features of vertex V and the edge-set E. The vertex features are high-level features based on zone-content and are validated against the low-level features of the edge-set for a possible merger. The problem then reduces to a graph-reduction problem and final vertices represent zones distinguishable by either distance or content from their neighboring zones. The sections below describe the two phases in detail.

3.2.2 Hypothesis Phase

The objective of the *hypothesis*-phase is to propose regions using low-level features. These low-level features are calculated based on dynamic adaptation of component separation and area features to local variations. Based on the decision on every edge, certain edges are marked *provisional* in plausible regions. The plausible regions and the edge-set ($G\{E, V\}$) are passed to *validation*-phase for further processing.

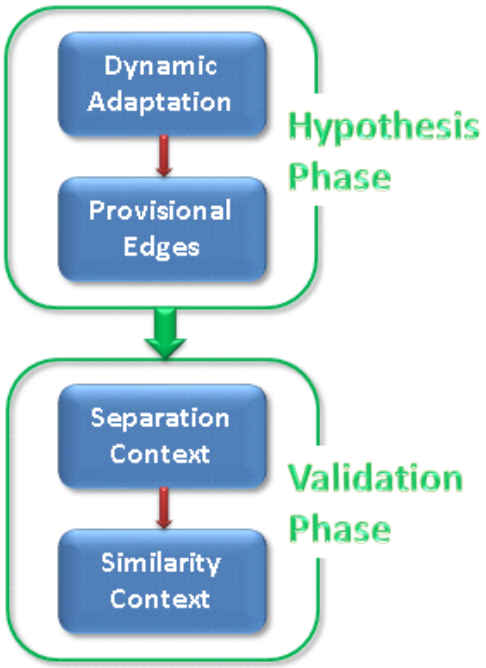


Figure 3.6: The two-phased Voronoi++ approach.

3.2.2.1 Dynamic Adaptation

In Kise's Voronoi based approach, global thresholds T_{d2} and T_a are determined statistically from the document's properties. T_{d2} is based on the second most frequent distance associated with the edges, which corresponds to the inter-word separation of prominent text-size in the document. T_{d2} and T_a have an inverse linear relationship with each other, hence neighbors with higher a_r need lower $d(E)$ to form an edge between them (as shown in Figure 3.10a). Typically T_a is chosen such that it forces an edge between components of drastically different sizes ($a_r = T_a$) separated by an epsilon ($d(E) \rightarrow 0$) distance. T_{d2} is determined based on second maxima and does get influenced by local peaks.

Since the global thresholds are determined statistically from a document, a

Voronoi based approach does fairly well when the document content is consistent as found in word, line and region separations of machine printed content. This criteria, however, is not always met. For example, in printed documents, the spacing between words or characters changes with font-size, text-direction and zone-type and in handwritten documents, the components belonging to the same zone may change their sizes drastically due to cursive nature of handwriting. In order to dynamically adapt these local variations in the size, orientation and distance of components within a page, the proposed approach either increases (grouping two or more regions) or decreases the thresholds (splitting a region into multiple zones). It also boosts Voronoi based approach's global threshold finding algorithm by using a more deterministic approach towards a more precise T_{d2}.

Increasing word-separation threshold: In order to avoid over-segmentation, spurious edges should be removed. We achieve this by increasing the word-separation threshold T_{d2} locally. This improves the following scenarios:

Avoiding over segmentation of larger text: Larger text has greater word-separation than the normal sized (more frequent) text that T_{d2} is based on. a_r still being close to 1, larger font text lines are often over-segmented into zones containing individual words or characters, as shown in Figure 3.7. We increase the word-separation threshold by a factor of the smaller of the two component's

size to the most frequent component size on the page.

$$T_{d2} = T_{d2} * fa_r$$
$$fa_r = \frac{average\ area\ of\ components}{most\ frequent\ area} \tag{3.5}$$

Figure 3.7: Image showing over segmentation of large fonts and dissimilar text sizes grouped together.

Angle-based merging: Voronoi weighs two components solely on the basis of their distance and area ratio. Missing components in degraded documents or a bad noise threshold can increase inter-character or inter-word gaps at certain places, thereby forcing edges, as shown in Figure 3.8. Such spurious edges can be removed by using the perceived text-direction which can be measured using the angle between two components. Docstrum uses this idea by describing each nearest-neighbor pair i,j by a 2-tuple $D_{ij}(d, \Phi)$ of distance d and angle ϕ between centroids of the two components. Since the mass-centroids of similar sized components may differ in ordinate positions due to concentration of mass at differing positions, we use the centroids of characters' bounding boxes. The overall text-direction in the document is measured by the most frequent angle α_{mode} between

a neighborhood pair. The closer an angle is between a neighboring pair to the frequent angle α_{mode}, the more likely it is to be a false gap. However, this is not a necessary condition. Text with two columns, for example, defy this concept. Hence, we calculate the deviation of a neighborhood angle α from α_{mode} as a Gaussian function and allow T_{d2} to be increased by a maximum factor of K. K should be chosen such that it restricts two-column merging.

$$dev = e^{-\frac{(\alpha-\alpha_{mode})^2}{2.\sigma^2}}.K$$

where σ is the deviation allowed (chosen value 30°), K is the factor limit with which T_{d2} can be increased. We chose K to be 0.5, allowing T_{d2} to be increased by a factor of 1.5.

$$T_{d2} = T_{d2} + T_{d2}.dev \tag{3.6}$$

Figure 3.8: Text-line direction can help merging the over-segmented regions as shown above.

Nearest Neighbor based merging: Some languages, such as Arabic, tend to have a lot of diacritics which, when handwritten, can vary significantly from their

ideal location. The area ratio of a diacritic with its corresponding consonant/vowel is often large and the original Voronoi based approach forces edges around such diacritics. Increasing the noise threshold, may cause diacritics to be eliminated as noise and hence they do not participate in edge formation. However, as stated in Section 3.1.1.2, choosing the correct noise-threshold is very difficult.

In our approach, any component smaller than the most frequent component size is associated to its nearest neighbor and the edge between them is removed. This removes the possibility of an edge between a diacritic and its corresponding vowel. Example results are shown in Figure 3.9.

Decreasing word-separation threshold: In order to avoid under-segmentation, removal of low-confident edges should be avoided. We achieve this by decreasing the word-separation threshold T_{d2} locally at appropriate places, segmenting a region into multi-zones. A scenario where this adaptation is necessary is as follows:

Separating dissimilar text sizes: Section headings tend to merge with the text if they are separated by distances similar to the inter-line distance of prominent text and their a_r value is low (Figure 3.7). Though a_r accounts for the difference in the areas of the two components, the relation between an increase in a_r and decrease in $d(E)$ is still linear as shown in Figure 3.10a. Hence, an increase in a_r, even by a factor of 4, for example, does not decrease $d(E)$ requirement for edge-formation by an appropriate amount. As per Equation (3.4), the minimum distance required between two components, with area-ratio as a_r, to form an edge

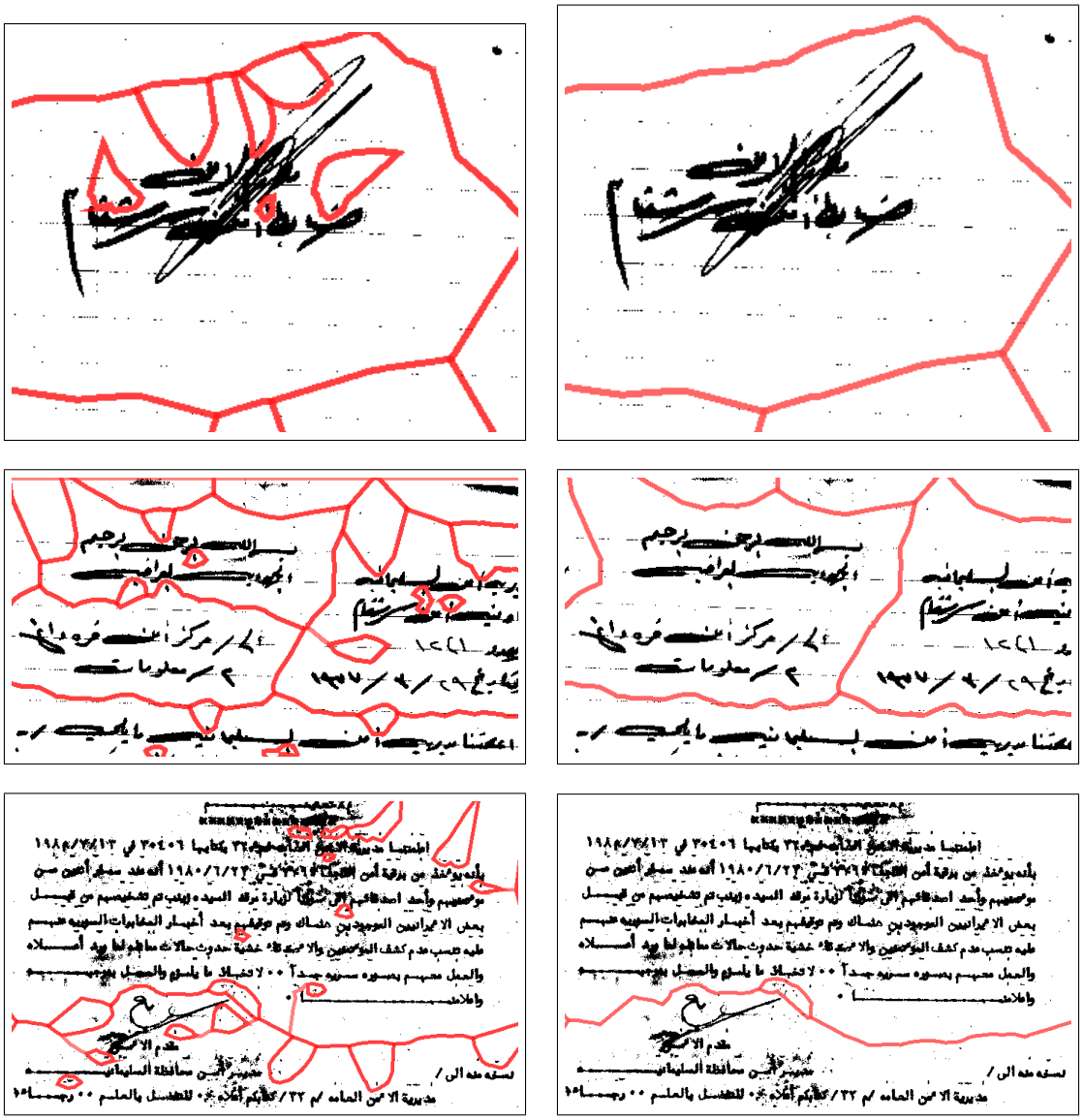


Figure 3.9: Illustration of results with and without NN associativity: The left side shows segmentation results without nearest-neighbor associativity and the right side shows the corresponding results with nearest-neighbor associativity.

is:

$$\frac{d_{min}}{T_{d2}} + \frac{a_r}{T_a} = 1$$
$$\implies d_{min} = \frac{T_{d2}}{T_a}(T_a - a_r) \tag{3.7}$$

If T_{d2} and T_a are constants, d_{min} decreases linearly with an increase in a_r.

We adapt the distance threshold T_{d2} dynamically with respect to the area-ratio between two components. Decreasing T_{d2} by a factor of a_r, improves the chances of edge-formation between dissimilar size components as follows:

$$T_{d2} = T_{d2} - T_{d2} * a_r/T_a \tag{3.8}$$

Substituting this in equation 4,

$$\frac{d(E)}{T_{d2} - T_{d2} * a_r/T_a} + \frac{a_r}{T_a} < 1$$

Solving, we get:

$$T_{d2}.a_r^2 - 2.T_{d2}.T_a.a_r - T_a^2.d(E) + T_{d2}.T_a^2 < 0 \tag{3.9}$$

This is an equation of a conic section of the form $Ax^2 + Bxy + Cy^2 + Dx + Ey + F = 0$. Since $B^2 - 4AC = 0$ and $C \neq 0$, the curve is indeed a parabola and demonstrates the parabolic relation. The new relation is depicted in the Figure 3.10b. Instead of a linear relation between T_{d2} and T_a, this dynamically adaptive parabolic relation [11] increases the probability of segmenting text of slightly varying sizes into separate zones. From Equation 3.9, the minimum distance required between

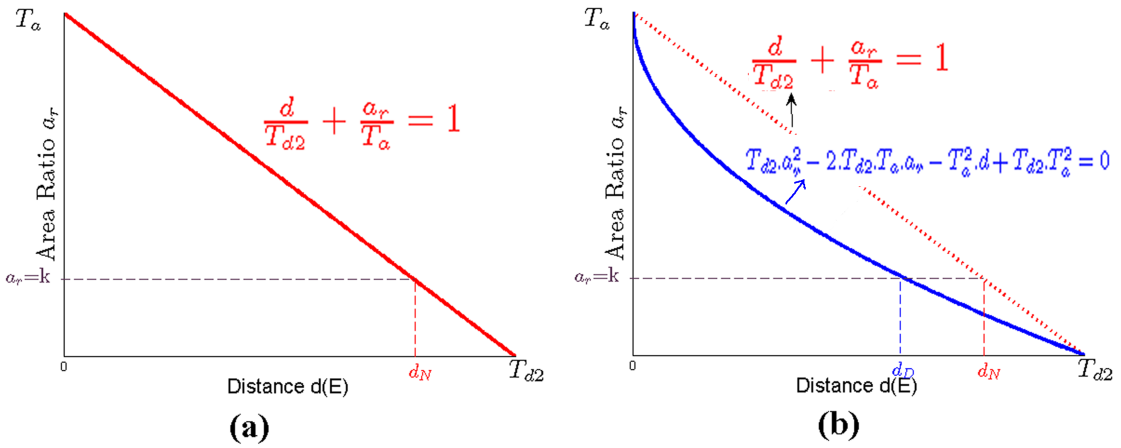


Figure 3.10: (a) Linear (b) Parabolic relation between inter-word distance and area ratio.

two components, with area-ratio as a_r, to form an edge is:

$$T_{d2}.a_r^2 - 2.T_{d2}.T_a.a_r - T_a^2.d'_{min} + T_{d2}.T_a^2 = 0$$

$$\implies T_a^2.d'_{min} = T_{d2}.a_r^2 - 2.T_{d2}.T_a.a_r + T_{d2}.T_a^2$$

$$\implies d'_{min} = \frac{T_{d2}}{T_a^2}(a_r^2 - 2T_a.a_r + T_a^2)$$

$$\implies d'_{min} = \frac{T_{d2}}{T_a^2}(T_a - a_r)^2 \tag{3.10}$$

Comparing Equations 3.7 and 3.10:

$$\frac{d'_{min}}{d_{min}} = \frac{\frac{T_{d2}}{T_a^2}(T_a - a_r)^2}{\frac{T_{d2}}{T_a}(T_a - a_r)}$$

$$\implies d'_{min} = d_{min}(1 - a_r/T_a) \tag{3.11}$$

This shows that with an increase in area-ratio, the dynamically adapted minimum edge-distance reduces as compared to the originally required minimum edge-distance, thereby forcing edges between dissimilar sized components. This also confirms the parabolic relation of the Equation 3.9.

Word Separation Threshold using Valley-Finding: Voronoi based approach statistically calculates T_{d1} and T_{d2} based on global layout and resolution. The two global peaks in the frequency distribution of distances associated with edges ($d(E)$) corresponds to T_{d1} and T_{d2} respectively, as shown in Figure 3.11. For such a raw frequency distribution $f_r(d)$, the smoothed distribution $f(d)$ is calculated using a window size ω, to get rid of local peaks [46]. However, it is not possible to choose

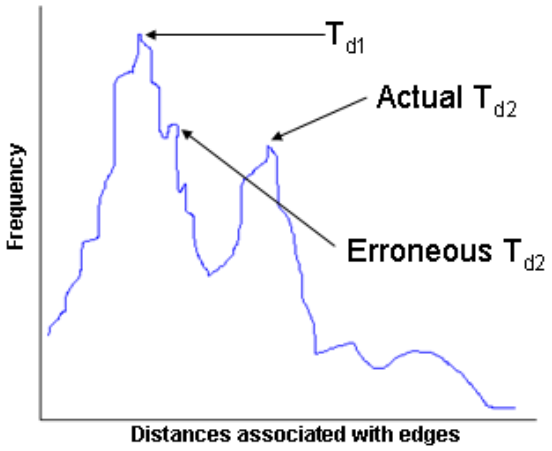


Figure 3.11: Frequency histogram of distances associated with edges.

ω correctly without knowing the real distance between the global peaks, which in turn depends on the correct value of ω. Hence, local extremas still remain in this process and subsequently affect global peak determination of T_{d1} and T_{d2}. This leads to local extremas being picked, as shown in Figure 3.11. The result is an over segmentation of the document.

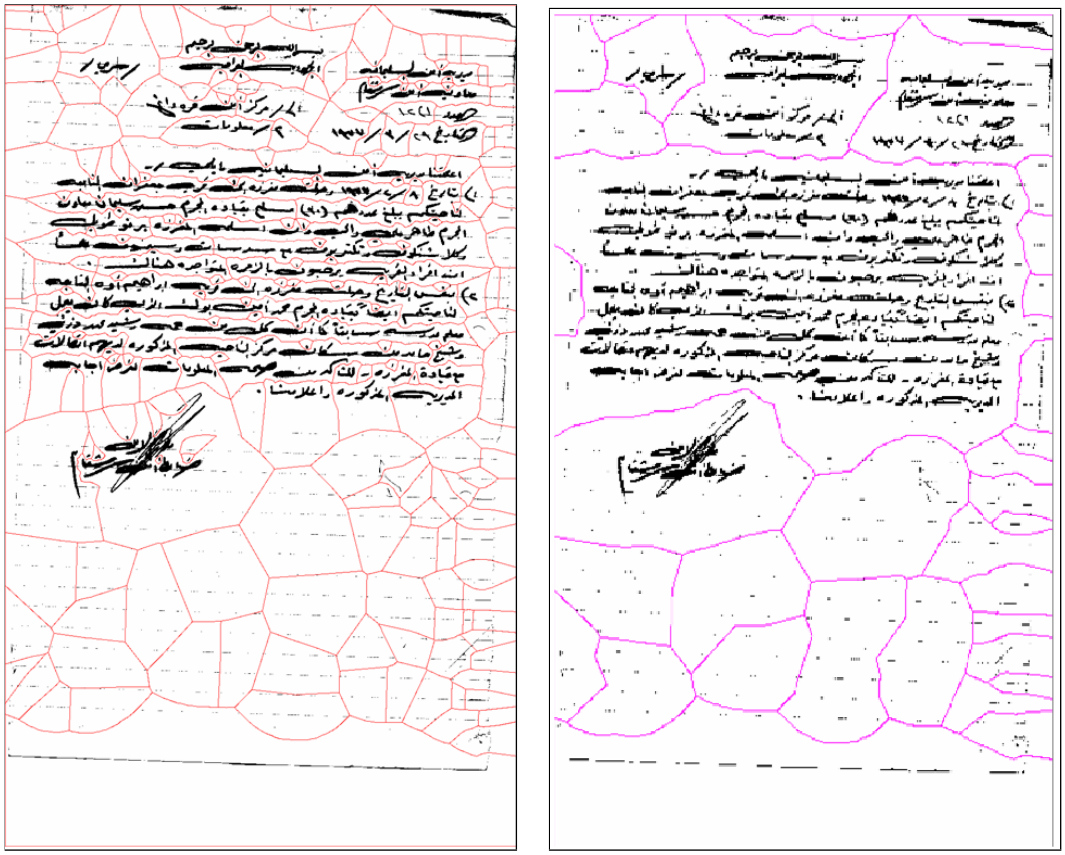
We use an improved 'valley' determination approach to divide the bimodal frequency graph in the middle of two global peaks. The maximas of each half

then correspond to the thresholds, bypassing the need for a smoothing function. In order to achieve this, we implemented an improved valley-finding algorithm by building a global cost function [13]. The goal is to avoid the common problems associated with typical valley-finding algorithms or reverse-gradient techniques. This bimodal frequency histogram can be treated as a valley, having foothills between two mountains. At each point on the histogram, we associate the cost of leaving the valley from either side. This is the minimum of the costs to reach either of the boundaries. When moving from a given point to either of the graph's boundaries, costs are incremented only when the histogram is increasing. The point with maximum cost is the point in the valley. This can be expressed as:

$$
\begin{aligned}
Cost_L(d) &= \sum_{k \le d, h(k-1) > h(k)} [h(k-1) - h(k)] \\
Cost_R(d) &= \sum_{k \ge d, h(k) < h(k+1)} [h(k+1) - h(k)] \\
Cost(d) &= \min(Cost_L(d), Cost_R(d)) \\
V_d &= \max_d(Cost(d))
\end{aligned} \tag{3.12}
$$

The largest peak on the right of V_d corresponds to T_{d2}. This avoids local peaks confusion and gives the best estimate of inter-word separation. The improvement in segmentation is depicted in Figure 3.12.

The Figure 3.13 shows how dynamic adaptation is a step towards our goal of Voronoi++ as it combines the features from both Voronoi-based and Docstrum approaches and improves upon them.



(a) (b)

Figure 3.12: (a) Page Segmentation using the original Voronoi approach (b) Page Segmentation after finding word separation thresholding using the improved valley-finding algorithm.

3.2.2.2 Provisional Edges

The approach described so far dynamically adapts to variations but makes only component-level decisions. Even though two components may trigger adaption due to their sizes, they may result from different content. Likewise, any faulty adaptation on a single edge may lead to the merging of two genuine zones. We mark certain edges as *provisional* in the *hypothesis* phase. The goal is to retain edges unless sufficient evidence is collected, label them as *provisional* and preserve them for subsequent validation. There are three scenarios where edges are marked

Algorithm	Separation: Distance	Separation: Direction	Content: Area Ratio	Content: Local Pattern	Context
RLSA	✔	✔			
X-Y Cuts	✔				
Whitespace Analysis Method	✔	✔			
Constrained Text-line Detection	✔	✔			
Docstrum	✔	✔			
Voronoi	✔		✔		
Dyn. Adap. Voronoi	✔	✔	✔		
Voronoi++	✔	✔	✔	✔	✔

Figure 3.13: Dynamically Adaptive Voronoi assures both Voronoi and Docstrum features.

provisional and require contextually-aware features (instead of component-level local features) to make the correct decisions.

Contradicting decisions: Voronoi++ avoids making binary decisions based at the component-level. Every edge e is given a confidence (γ^G) by:

$$\gamma^G = \frac{d(E)}{T^G_{d2}} + \frac{a_r(E)}{T_a} \tag{3.13}$$

where the word-separation threshold T^G_{d2} is a globally determined threshold before dynamic adaptation [11]. As in the Voronoi based approach, an edge is preserved if $\gamma^G > 1$. After dynamic adaptation, the word-separation threshold T^D_{d2} is determined and γ^D is calculated as described above. If after dynamic adaptation, the decision on an edge is reversed (i.e. $\gamma^G > 1$ & $\gamma^D < 1$ *or* $\gamma^G < 1$ & $\gamma^D > 1$), it is preserved and tagged as *provisional*. This is because the decision made dynamically based on local properties is kept as a hypothesis for validation. This avoids

merging of zones due to incorrect adaptation. Figure 3.14 illustrates the scenarios where this can happen.

Inter-word threshold (T_{d2}): The two peaks in the frequency distribution of distances associated with the edges correspond to inter-character and inter-word gaps. v_1 corresponds to the distance for the first peak and v_2 corresponds to the distance for the second peak. The aim of determining the correct T_{d2} is to delete all edges separating words or lines. The T_{d2} determination poses two challenges. The first challenge is to find the correct v_2. In Section 3.2.2.1 we proposed a valley finding approach to better determine v_2. The second challenge is to determine T_{d2} from v_2. The value of v_2 is inappropriate as T_{d2}, because some of these edges (between lines and words) have values slightly larger than v_2, as shown in Figure 3.15. Kise et. al. [46] add a margin to v_2 to determine T_{d2} as follows:

$$T_{d2} > v_2 \tag{3.14}$$

$$f(T_{d2}) = t.f(v_2) \tag{3.15}$$

where t is the parameter for controlling the margin. Since $f(d)$ takes discrete values, linear interpolation is applied to obtain T_{d2} which satisfies Equation 3.15. If multiple values of T_{d2} are obtained, the smallest one is used. Since t is more of a subjective parameter, an exact determination of T_{d2} is not possible. The edges with distances between the smallest interpolated value of T_{d2} and the dip after v_2 are equally likely to be removable edges (highlighted in Figure 3.15). These *weak* edges are also marked as *provisional* and kept under supervision for removal in

Figure 3.14: A and B depict increase in T_{d2} based on larger components, due to which dissimilar zones have been merged together. C depicts large components formed due to touching-characters *(be, Dec)*. This in turn increases T_{d2} and removes the edge, leading to the fusion of two columns into a single zone.

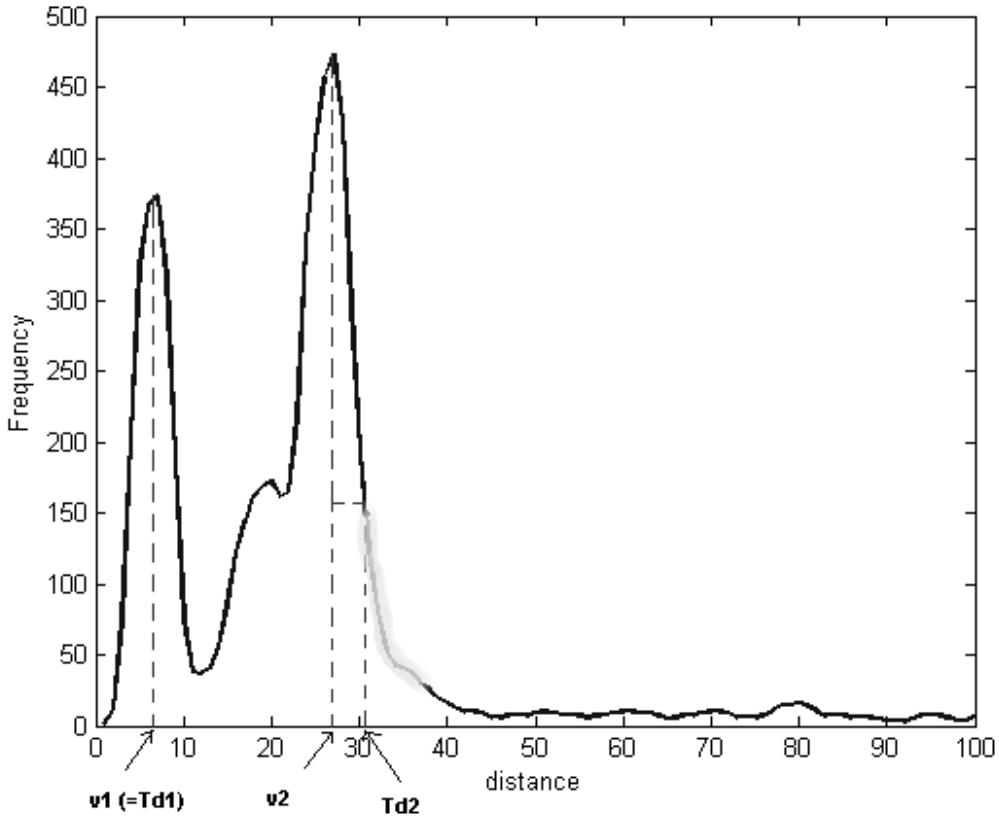
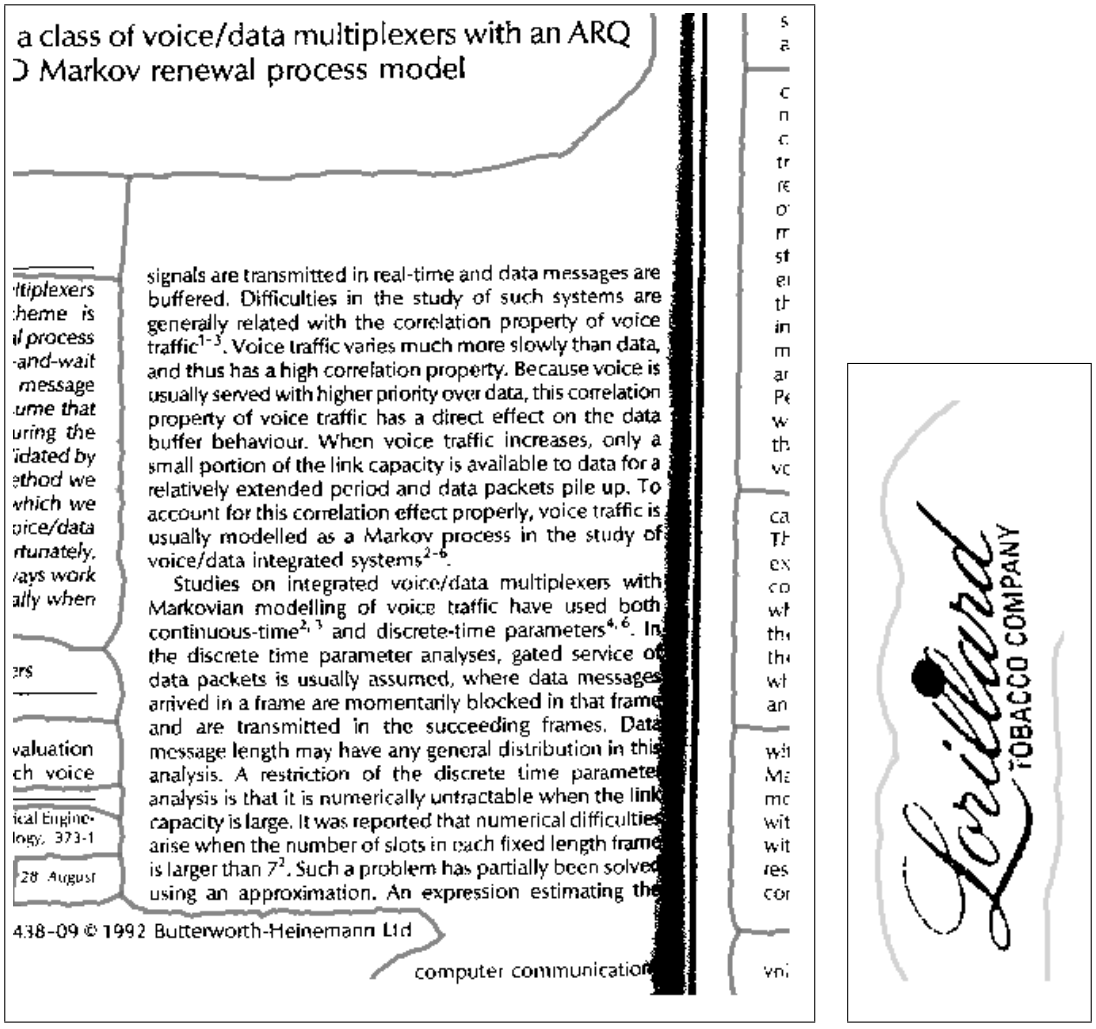


Figure 3.15: Highlighted area shows edges with slightly larger distances than T_{d2} and may be equally likely to be removed.

the *validation*-phase.

Inter-character threshold (T_{d1}): Both Voronoi and the proposed dynamically adaptive scheme remove edges between components separated by inter-character distances (Equation 3.4) without adaptation. This condition helps prevent edges within the same zone (i.e. preventing over-segmentation) but may also lead to removal of edges between distinct zones separated by distances less than inter-character thresholds as shown in Figure 3.16. In the *hypothesis*-phase, we do not



(a) (b)

Figure 3.16: Components belonging to different zones are merged if separated by less than the character-separation threshold.

want to merge distinct zones at the cost of over-segmentation, so this condition is removed from segmentation.

Any edges that would have been removed by dynamic adaptation are now preserved with a *provisional* label along with the edges which were 'saved' from removal due to adaptation. This results in more edges when compared to the original Voronoi or the proposed dynamically adaptive scheme (Section 3.2.2.1).

The *hypothesis*-phase associates the following low-level features with each edge:

1. distance $d(E)$
2. confidence γ^G
3. isProvisional (*bool*)

3.2.3 Validation Phase: Considering Context

The zones obtained after the *hypothesis* phase are separated by edges containing low-level inter-component relationship features. The goal of this phase is to merge zones based on their content similarity as well as the low-level features separating them. It is important to understand that low-level separation features are as important as high-level content features. While two adjacent zones may have similar content, they may still belong to physically separate zones (e.g. two adjacent columns on a page). The challenge in segmentation is to determine the correct separation based on local neighborhood patterns. A contextually aware system has a better chance of determining that instead of a system based on only low-level features. The *validation* phase builds context using low-level distance based features (called Separation context) and high-level zonal content (called Similarity context). Two zones are merged only if they satisfy the criteria in both contexts.

3.2.3.1 Separation Context

Instead of merging two zones based on a single provisional edge, separation-context determines the percentage of provisional edges separating them in their edge-set E. However, not all non-provisional edges are of interest for the separation context. Those non-provisional edges whose confidences are limited by a factor of maximum provisional edge confidence are labeled critical edges: $\gamma_c < K * \max_f(\gamma_f)$ where K is a constant chosen to be 2 and γ_f is the confidence of a provisional edge f. All other stronger edges are non-critical edges which do not contribute in decision making. Two zones are said to be a single zone candidate

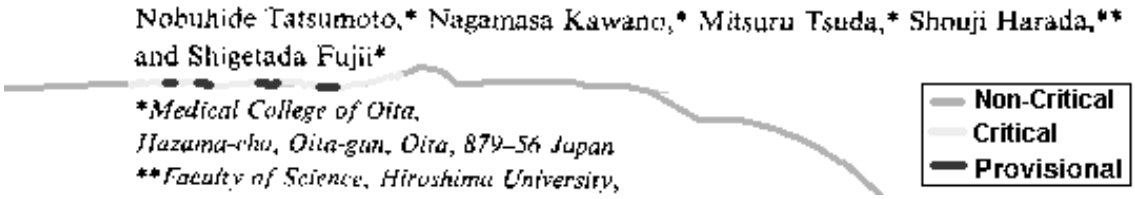


Figure 3.17: Non-critical, critical and provisional edges separating two zones.

if the ratio of provisional edges to critical edges is above a threshold:

$$\frac{|ProvisionalSet|}{|CriticalSet|} > \tau \tag{3.16}$$

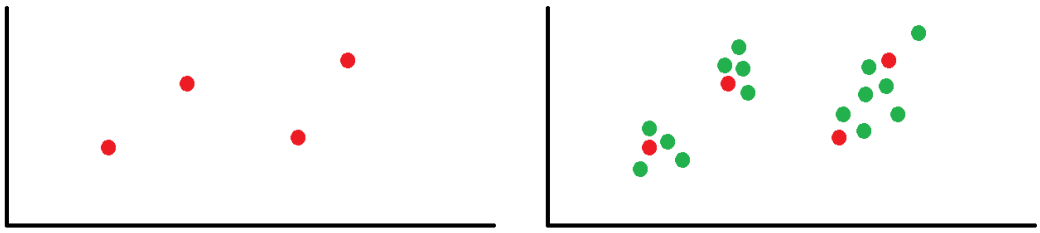
where τ is chosen to be 0.5 based on empirical studies. Figure 3.17 shows provisional, critical and non-critical edges separating two zones.

3.2.3.2 Similarity Context

All candidate zones from the *separation* context are fed into a feature extraction module. Broadly there are two types of feature extraction techniques. One

type tries to represent the structures [44, 76, 52, 34] and another type tries to represent the textures [66, 93]. For object detection, the former set of techniques have been quite successful and are generally rotation and size invariant. For zone classification, however, zones may have the same textual content, which may differ in size, type or rotation. Hence, texture based features are typically required. We use a combination of run-length based features and spatial features derived from foreground and background pixels separately [8]. These signature-like features encode pixel spatial distribution information and amount of local image texture (contrast).

Each extracted d-dimensional feature vector is visualized as a data point in d-dimensional space. If N is the total number of points, the goal is to group them into c clusters, where $c \epsilon [1, N]$. As with any clustering problem, determining the correct c is not a trivial problem. A deeper analysis reveals that this data is not *flat* (i.e. it contains hierarchy levels of varying within-cluster distances). While a figure zone and small-text zone may reveal feature vectors occupying distant places in the d-dimensional space, features of bold and italics text may lie very close to each other. This requires a more sophisticated hierarchical clustering approach with varying within-cluster thresholds. However, if for every feature vector, an approximate distance to a similar feature vector was known beforehand, the problem reduces to a semi-supervised clustering. In addition, if we know the approximate area spanned by similar feature vectors in d-dimensional space, we have stronger evidence of similarity for every point. Figure 3.18 demonstrates the concept.



(a) (b)

Figure 3.18: (a) shows 4 feature-vectors spaced in 2-D. All vectors are separated from each other by nearly equal distances. (b) shows few similar feature-vectors to each vector. The clusters become apparent.

To achieve this, we divide each zone into two parts along the length of its bounding box and N tiles of size ω X ω where ω is the most frequent component's size (determined from dynamic adaptation). Each zone, along with its 2 halves and N tiles, is sent to the feature extraction module, generating $N + 3$ feature points. These features encode information at various levels and positions in each zone. Each zone's feature vector in d-space along with its *similar* vectors (from two halves and N tiles) form a set of points called a supervised-neighborhood κ. The problem now reduces to a semi-supervised clustering problem, where for each neighboring pair of zones i and j, κ_i is compared with κ_j. If they form two distinct clusters in d-space, the zone merge is said to have been *prevented*, else they are merged. This is not an optimization problem as we do not intend to form two clusters out of two sets of points. Instead, the goal is to verify if the two sets of points form two clusters.

Since the classification is unsupervised and the number of feature vectors (of two zones under validation) are typically less than the dimensionality d of feature space, the features of the zones can not be mapped to a more discriminative (or

reduced) feature space using approaches like PCA, LDA or PLS [8]. Instead, if the distance between the means (μ_i) of two supervised-neighborhoods is greater than the sum of their average radii (R_i), the neighborhoods are said to represent distinct zones, otherwise their zones are merged. In the case of a merge, the new supervised-neighborhood of the merged zone contains feature vectors from both zones, thereby encompassing more context.

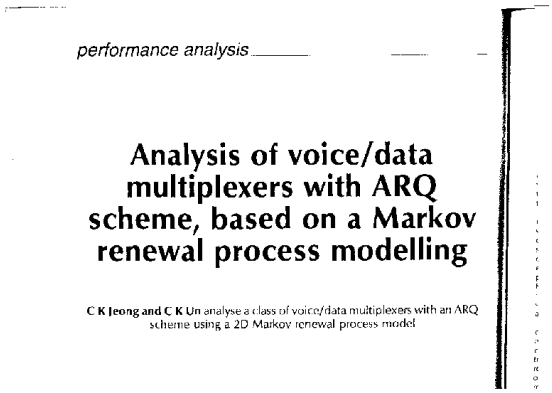
$$
\begin{aligned}
R_1 &= \sum_{x \epsilon \kappa_1} (\vec{x} - \vec{\mu_1}) / ||\kappa_1|| \\
R_2 &= \sum_{x \epsilon \kappa_2} (\vec{x} - \vec{\mu_2}) / ||\kappa_2|| \\
D &= \vec{\mu_1} - \vec{\mu_2} \\
if\ R_1 &+ R_2 > D,\ MERGE
\end{aligned}
\tag{3.17}
$$

Figure 3.19 shows a document image at each stage of the Voronoi++ approach.

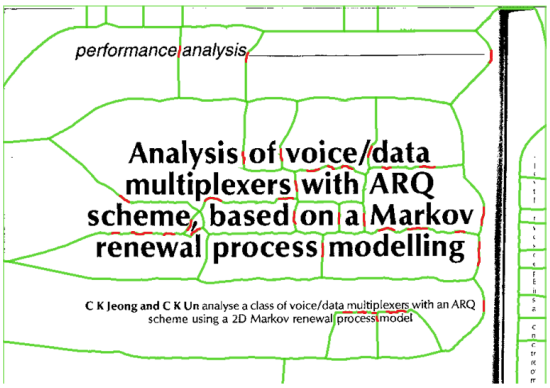
3.2.4 Evaluation

3.2.4.1 Metrics

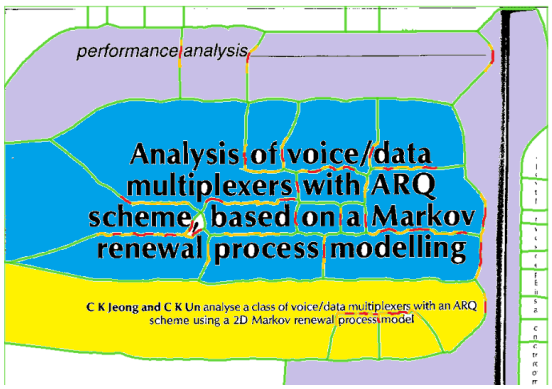
Evaluation of page segmentation algorithms is an important but controversial step for the document analysis community. The primary reason is that it is difficult to provide a deterministic way to divide a page into zones that everyone will agree with and that is appropriate for all tasks, especially for complex documents. While it is evident that two zones with different styles should be separated, splitting a text-zone between text-lines, for example, is arguably acceptable. A



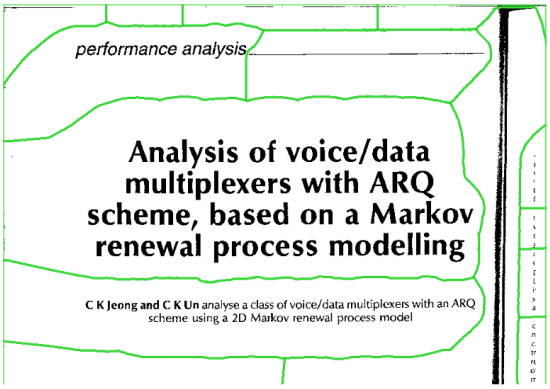
(a) Original document image



(b) Over-segmented zones with provisional edges marked in red



(c) Proposed regions after Separation Context, with critical edges in yellow



(d) Final segmentation after Similarity Context

Figure 3.19: Context-aware phases of Voronoi++.

two-column document page will have at least two zones, but each column can be further split at its paragraph or line breaks without affecting reading order or comprehension. In order to avoid this ambiguity, the accuracy of page segmentation algorithms is often calculated as the percentage of ground-truth text-lines contained correctly within result zones without split, merge or miss errors [81, 58]. The drawback of this approach, however, is that if the segmentation algorithm outputs the whole page as one segment, the split and missed errors disappear. As shown in Figure 3.20, the result zones containing complete text-lines from different zones are not penalized. In [11] we proposed a zone-based evaluation method where we compare ground-truth zones with result zones. A result zone is said to be detected, if its foreground pixels overlap those of ground-truth above a user specified percentage. This is a much stricter evaluation scheme in terms of zone detection. However, as stated earlier, this method will not tolerate *valid* over-segmentations (at paragraph or line breaks).

A more complete evaluation scheme is now used, which involves ground-truth at both zone and text-line level. After ground-truthing text-lines, lines are grouped into the largest possible distinct zones. While it remains a zone-based evaluation strategy, valid (along-text) zone splits are not penalized. This *split* option is configurable and the evaluation tool is used for both *split* (lenient) and *no-split* (strict) options [78]. Due to the current lack of ground-truth data at text-line level, the *split* option allows zone-splits irrespective of their validity. Another option, called the *ignore* option, ignores all those result zones whose precision and recall are zero, i.e. if they don't overlap with any ground-truth

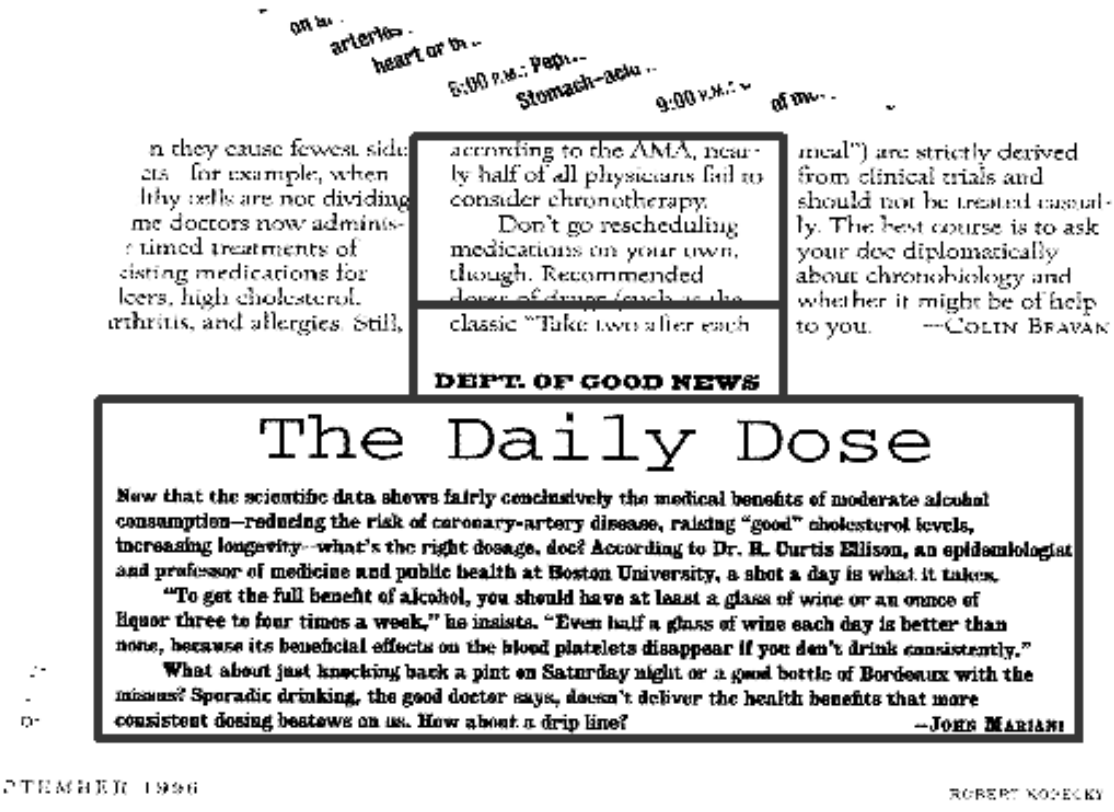


Figure 3.20: Result zone covering two distinct zones is not penalized using line-based evaluation.

zone. This avoids penalizing the segmentation algorithms for producing zones at unannotated regions such as marginal noise and text-borders (partial text-regions from an adjacent page in case of book scanning).

In order to gauge the effects of the strict and more lenient evaluations of our approach, we use 3 kinds of evaluation schemes based on the options in our evaluation tool [77] (Figure 3.21, Table 3.1). The first evaluation scheme E_1 does not penalize the algorithms for producing zones at unannotated regions (*ignore*-option: true) while the second evaluation E_2 does. The third evaluation scheme E_3 allows zone splitting within a ground-truth zone (*split*-option: true). Precision,

(a) Evaluation Scheme E_1

(b) Evaluation Scheme E_2

(c) Evaluation Scheme E_3

Figure 3.21: 3 metrics are used for comparing various page segmentation algorithms. The rectangular blocks show ground-truthed zones and result zones are depicted as polygons. The noise regions are not ground-truthed.

Table 3.1: Evaluation Schemes

Evaluation Scheme	**Description**
E_1	Ignores segmentation results on unannotated regions
E_2	Penalizes segmentation results on unannotated regions
E_3	Allows zone splitting within annotated regions

Recall and F1-scores are calculated as follows:

$$Precision = \frac{True\ Positives\ (TP)}{True\ Positives\ (TP) + False\ Positives\ (FP)} \tag{3.18}$$

$$Recall = \frac{True\ Positives\ (TP)}{True\ Positives\ (TP) + False\ Negatives\ (FN)} \tag{3.19}$$

$$F1 = \frac{Precision + Recall}{2} \tag{3.20}$$

The ground-truth and result files (containing polygonal zones) follow the GEDI XML format specification [53].

In order to separately evaluate the improvements due to the proposed dynamic adaptation scheme, without context (Section 3.2.2.1) and with the complete two-phased context-aware scheme, we compare the final Voronoi++ to both the original Voronoi system and the Voronoi++ (without context).

3.2.4.2 Datasets

We evaluate the original Voronoi approach and our Voronoi++ approach against 3 datasets using different configurations of the region-based evaluation scheme. The 3 datasets (Table 3.2) are:

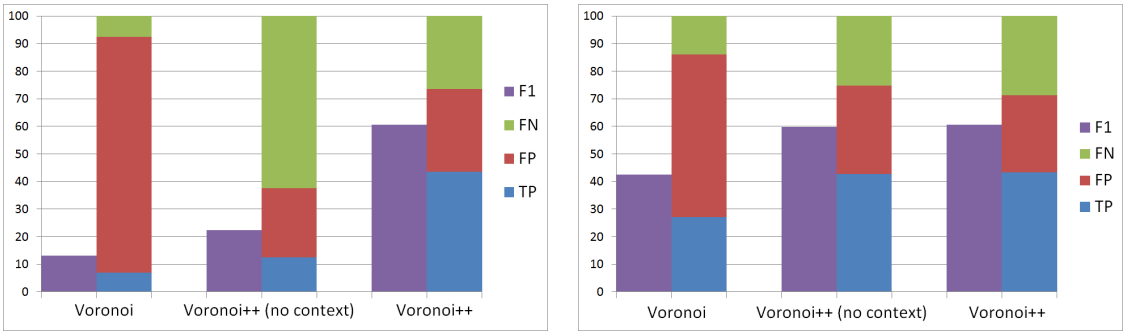
Table 3.2: Datasets

Dataset	**Language**	**Content-Type**	**Zone geometry**	**Noise**
D_1	Arabic	Handwritten	Polygonal	Clutter, rule-lines, salt-n-pepper, stray-marks
D_2	English	Machine-print	Rectangular	Clutter, salt-n-pepper
D_3	English	Machine-print and Handwritten	Polygonal	Degraded, clutter, salt-n-pepper

1. Arabic Handwritten (D_1): This dataset consists of handwritten Arabic text, stamps, logos and figures with noise in the form of stray marks, clutter and salt-n-pepper. The zones were polygonal.

2. English Machine-Print (D_2): The second dataset is from University of Washington III(UW-III) database [39]. It contains 10 different zone types - chemical drawing, small text, symbols, drawing, halftone, logo or seal, map, math, table and large text. The selected documents were primarily scanned publications or journals with rectangular zones.

3. Complex English (D_3): The third dataset consists of highly degraded and noisy English documents. The documents consist of forms, handwriting annotated printed text, tabular columns, letters and memos. The zones were primarily polygonal.



(a) Arabic Results

(b) English Results

Figure 3.22: Accuracy comparison for Arabic and English datasets. FN = False Negatives, FP = False Positives, TP = True Positives

3.2.4.3 Experiments and Results

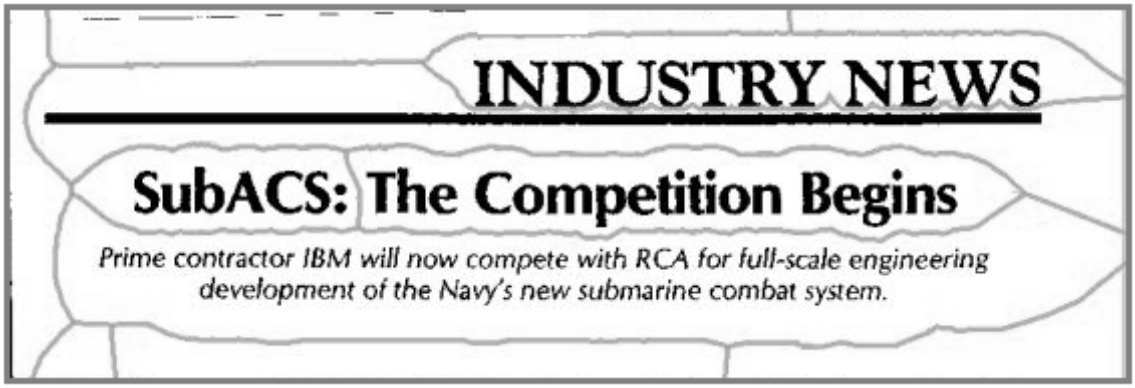
Scripts Comparison: The accuracy results using 50 documents from each dataset under evaluation scheme are shown in Figure 3.22. For the Arabic dataset evaluation, Voronoi++ improves the accuracy by over 400 percent, underlining the limitation and room for improvement of state-of-the-art algorithms for handwritten non-Latin scripts. Using both content-based and context-aware features, Voronoi++ further triples the overall accuracy (precision) over its context-unaware counterpart for Arabic dataset. It also achieves an improvement of 18% for English datasets.

Figure 3.23 shows example results of Voronoi, Voronoi++ (no context) and Voronoi++ from the English D_2 set while Table 3.3 shows samples from the Arabic D_1 set.

Metrics Comparison: We compare a subset of 100 English documents (50 from each English dataset D_2 and D_3) across the three defined metric schemes (Section 3.2.4.1) to evaluate the consistency of Voronoi++ over the traditional Voronoi

(a) Voronoi

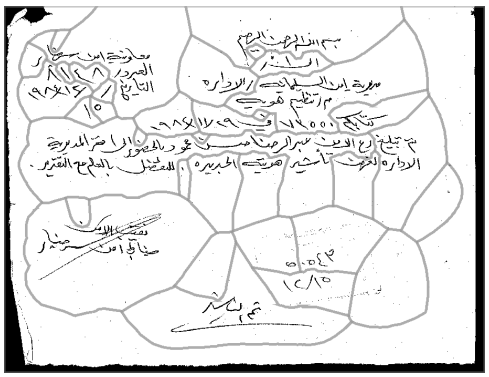
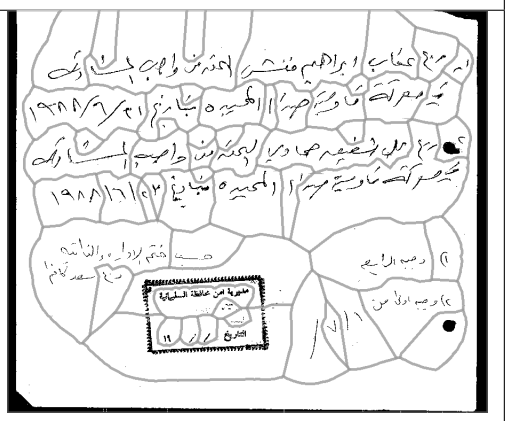
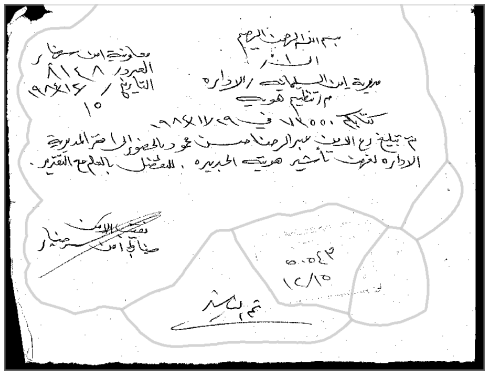
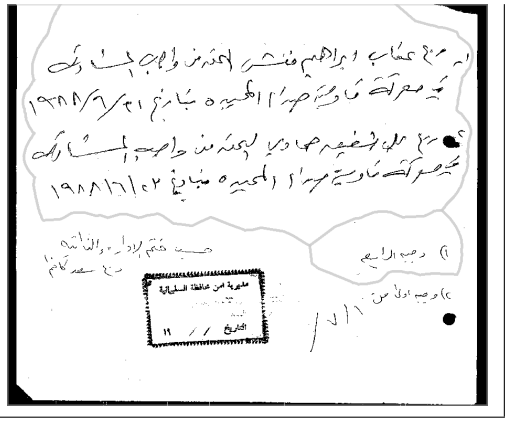
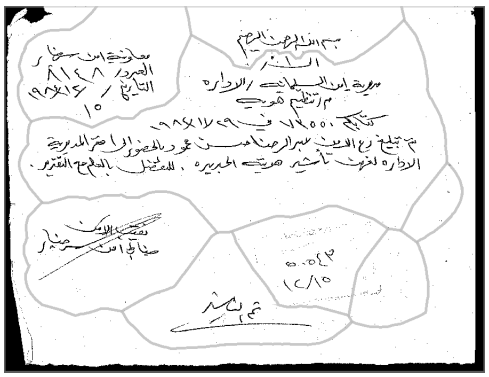
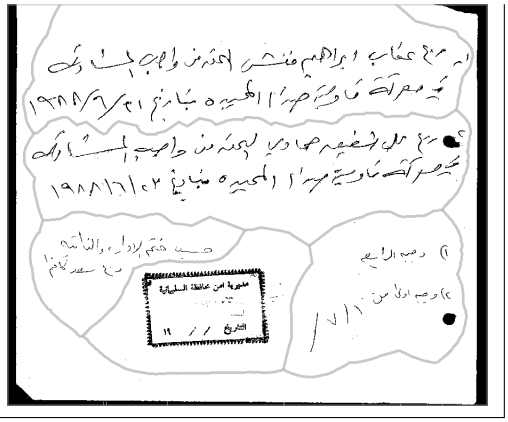
(b) Voronoi++ (no context)



(c) Voronoi++

Figure 3.23: Illustration of results of various segmentation approaches on printed English dataset.

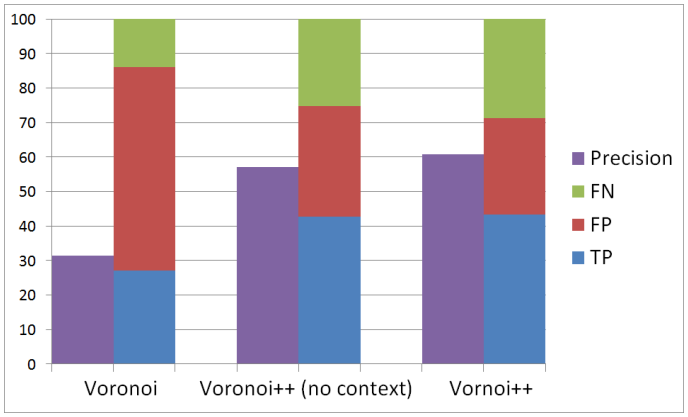
Table 3.3: Illustration of results of various approaches on handwritten Arabic dataset.

	Document A	Document B
Voronoi		
Voronoi++ (no context)		
Voronoi++		

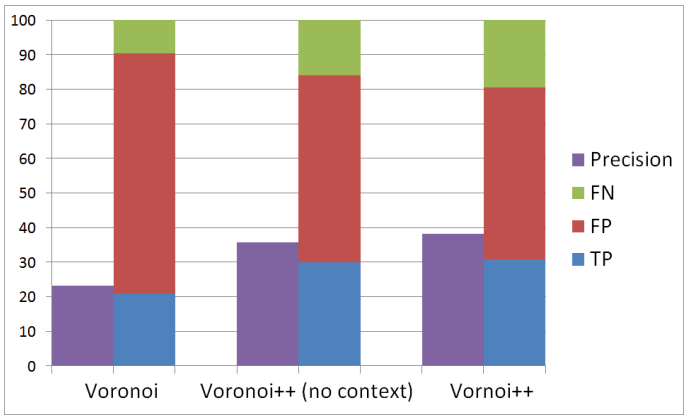
approach (Figure 3.24). The false-positive zones are filtered based on the *split* and *ignore* flags of the schemes (Figure 3.21). Precisions are compared across schemes along with the percentage of true-positives, false-positives and false-negatives zones returned. Voronoi++ consistently performs better in all the evaluation schemes. Figure 3.25 also depicts Voronoi++ reduces over-segmentation (better Recall) over Voronoi and, on an average, reduces the number of zones returned to 50% for English datasets.

Parameter Sensitivity: In Section 3.1.1.2, we illustrated how tuning parameters effect page segmentation and the challenges involved in choosing the correct noise threshold for a particular dataset. One of the goals of Voronoi++ was to remove the dependency on such input parameters and adapt to any dataset with minimal user intervention or experimentation. In order to verify this, we evaluated and compared both Voronoi and Voronoi++ over a series of noise-thresholds on Arabic and English datasets. Both datasets had different resolution and component sizes. As illustrated in Figure 3.26, accuracy of Voronoi page segmentation algorithm peaks at different noise thresholds for Arabic and English datasets (at thresholds 100 and 20 respectively). On the contrary, Voronoi++ performed best at a consistent noise threshold across datasets (threshold 20).

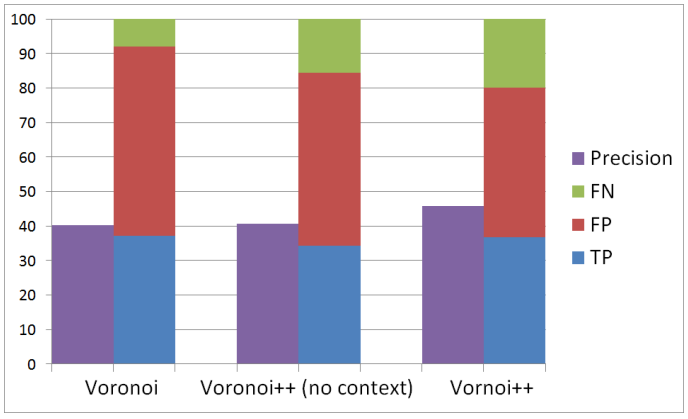
Overall Results: Voronoi++ is also evaluated against Voronoi on a larger set of 350 randomly selected documents from all the datasets (D_1, D_2 and D_3) using the first evaluation scheme E_1. The overall increase in accuracy is by 74% and



(a) Results using E_1 scheme



(b) Results using E_2 scheme



(c) Results using E_3 scheme

Figure 3.24: Metrics comparison for English datasets. Voronoi++ improves accuracy consistently. FN = False Negatives, FP = False Positives, TP = True Positives

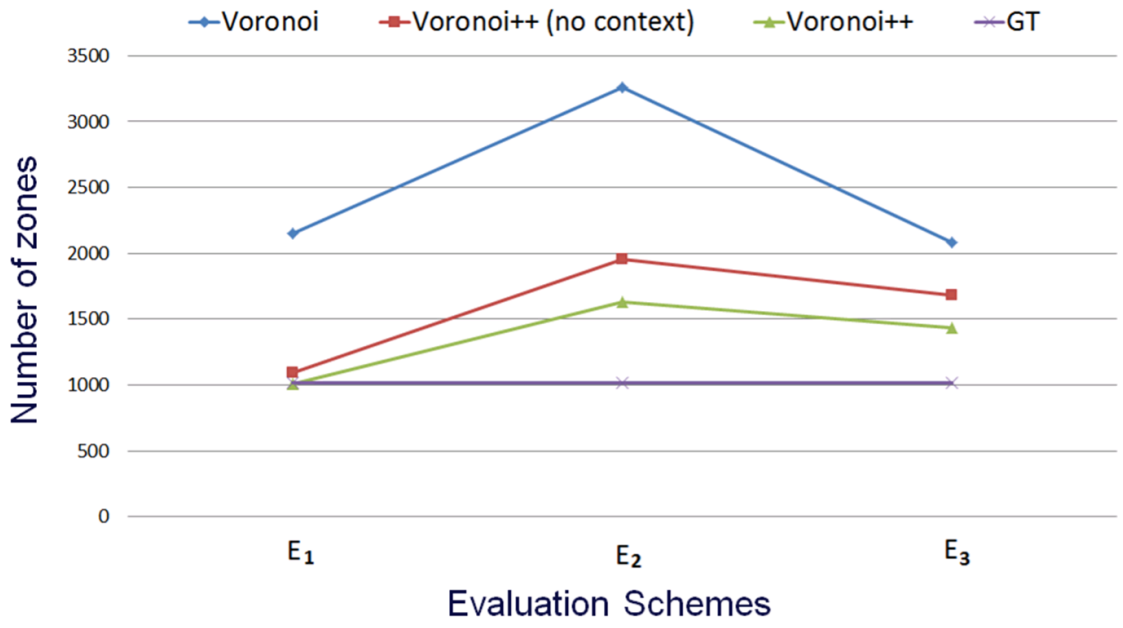


Figure 3.25: Number of zones returned from Voronoi++ are closest to those in ground-truth across different evaluation metrics.

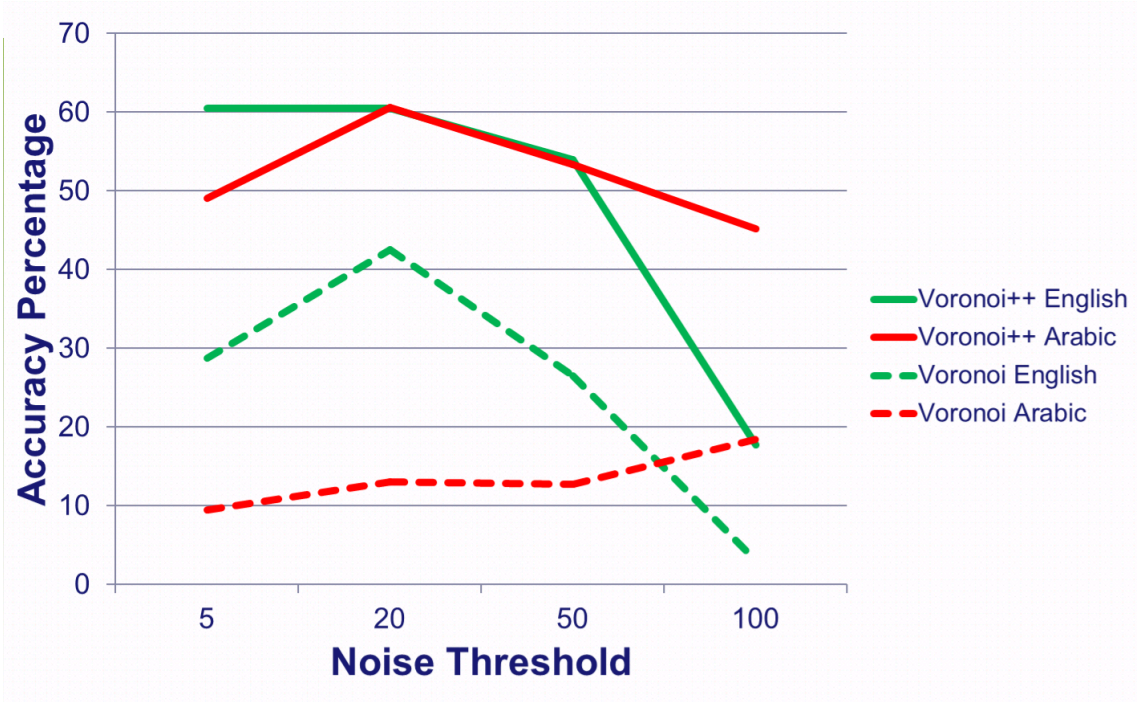


Figure 3.26: Voronoi++ performs best at a consistent noise threshold (=20) across Arabic and English datasets, whereas Voronoi's accuracy peaks at different thresholds for different datasets.

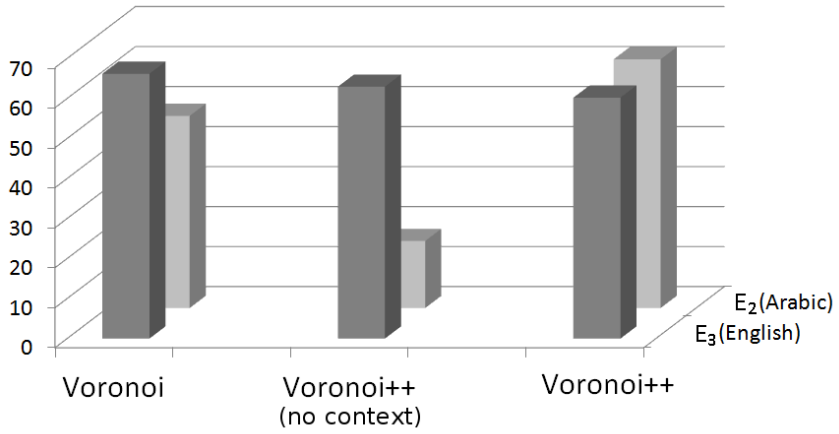
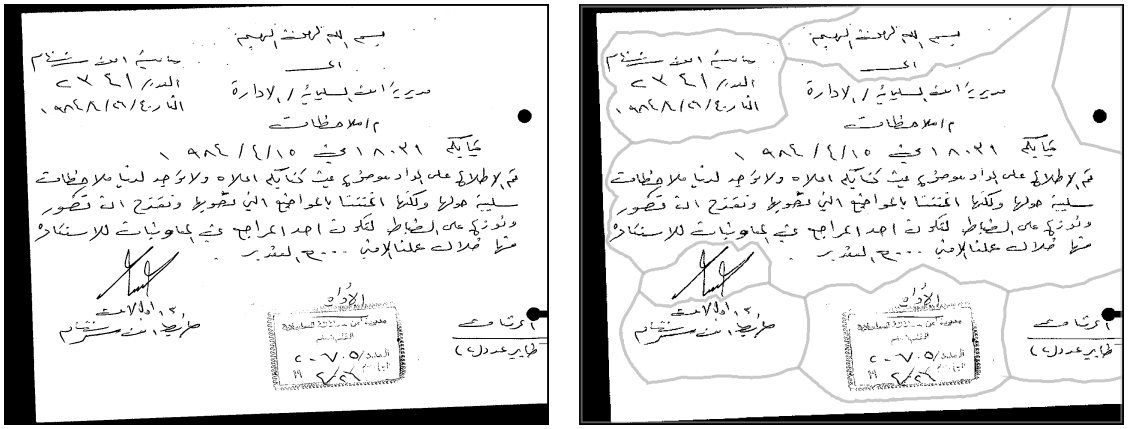


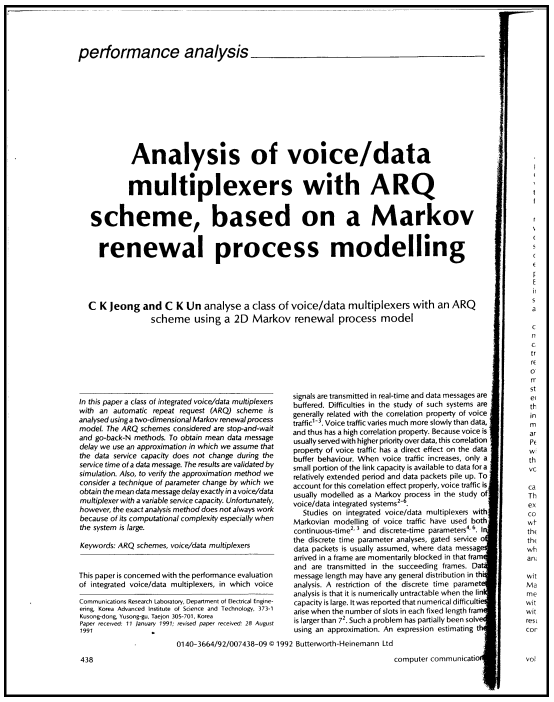
Figure 3.27: Recall of all approaches against different evaluation schemes.



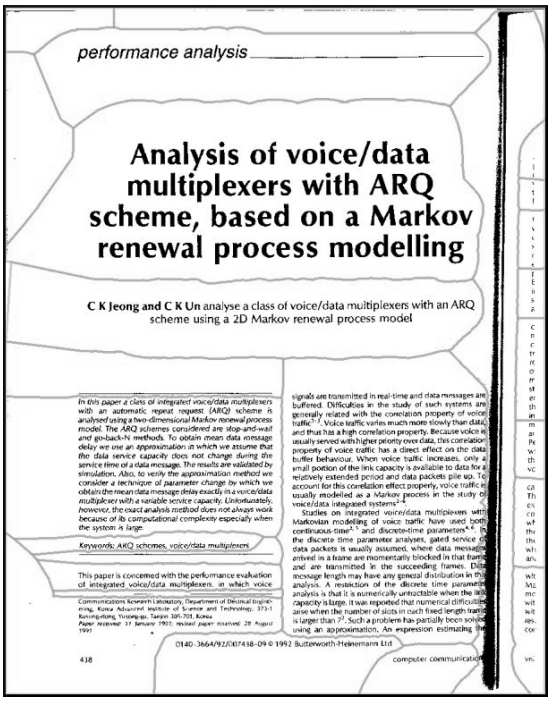
(a) D_1: Document Image (b) D_1: Voronoi++ Segmentation

Figure 3.28: Voronoi++ results on various datasets.

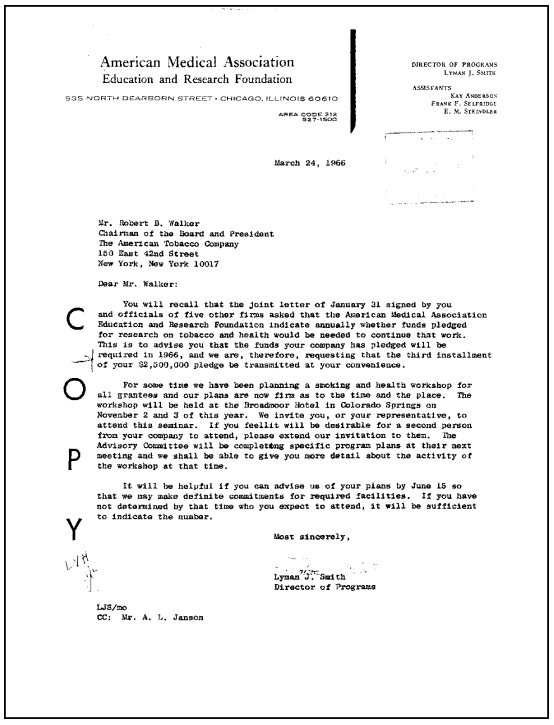
the number of false-positives reduced by 73%. Figure 3.27 illustrates the recall of Voronoi++ is nearly equal to or better than Voronoi. Figure 3.28 shows sample documents from each dataset and their segmentation using the Voronoi++ approach.



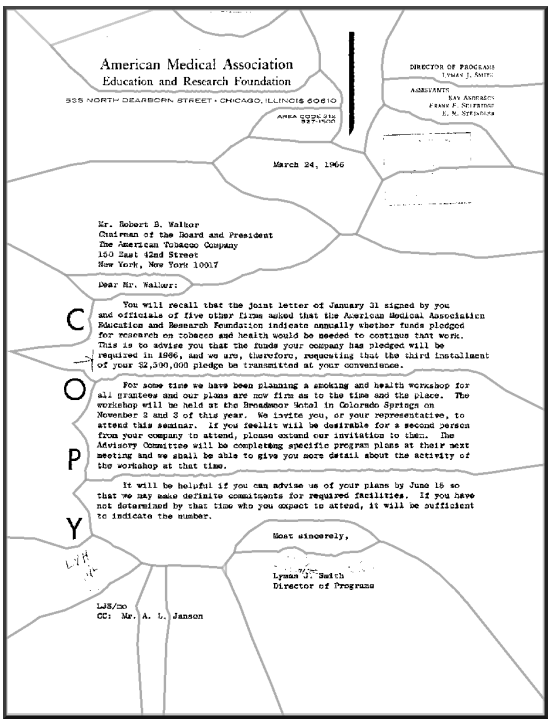
(c) D_2: Document Image



(d) D_2: Voronoi++ Segmentation



(e) D_3: Document Image



(f) D_3: Voronoi++ Segmentation

Figure 3.28: Voronoi++ results on various datasets.

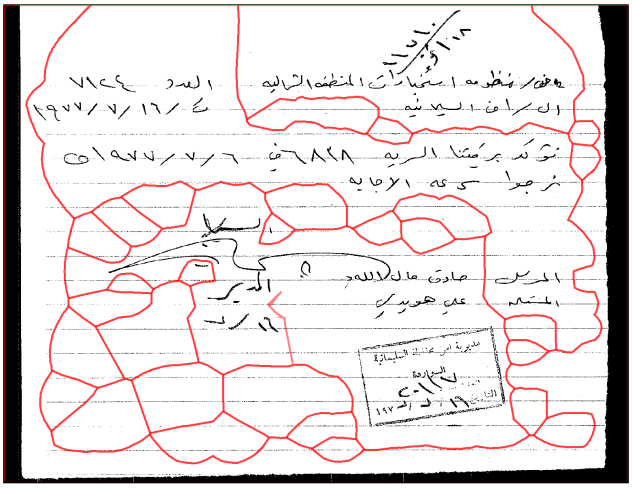
3.2.4.4 Error Analysis

Voronoi++ still makes some errors, and our analysis suggests that they can be roughly categorized as follows:

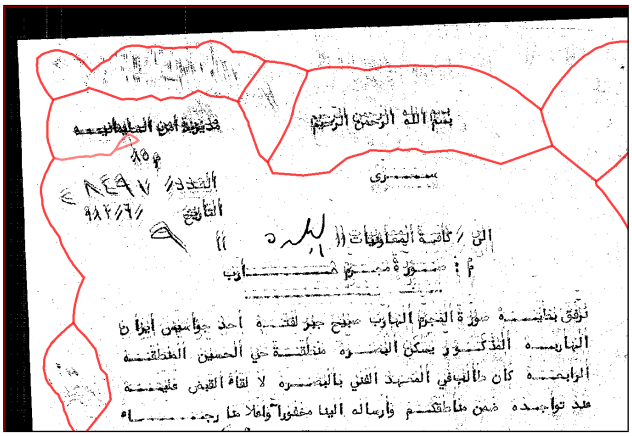
Document Quality: Rule-lines, salt-n-pepper noise and severly broken characters either merge dissimlar zones or lead to over-segmentation (Figure 3.29). A cleaner document has a much better chance of improved segmentation using Voronoi++.

Ambiguous Ground-Truth: A part of low accuracy can be attributed to inappropriate ground-truthing for evaluating component-based (white-space based) segmentation approaches. Overlapped and nested zones along with zones containing physically separated regions (Figure 3.30) can be seen in our datasets.

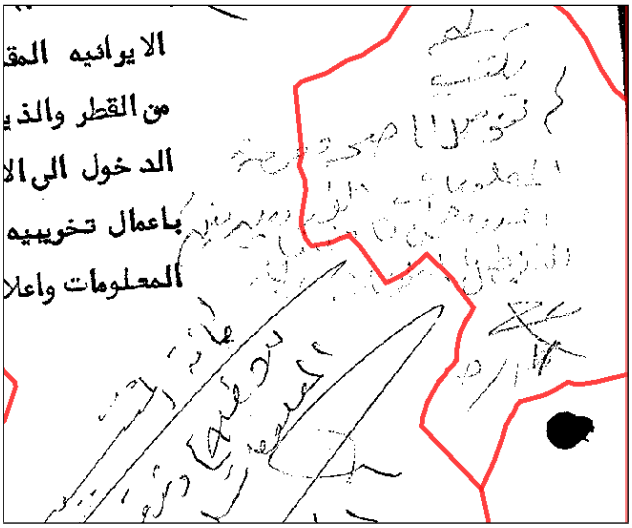
Tables and Charts: Tables and Charts are regions which require special definitions of similarity and separation and contradict those for textual or image-based regions. Tables are composed of similar columns, separated by considerable white-spaces while charts are composed of dissimilar regions like axes, numbers and the depicted data. While Voronoi++ outperforms Voronoi in grouping similar regions in these two categories, it fails in understanding the bigger context of Tables and Charts as shown in Figure 3.31.



(a) Rule lines prohibit edge-creation between different zones



(b) Salt-n-pepper noise tends to produce regions around dense noise



(c) Severely broken characters convey incorrect component-size leading to faulty segmentation

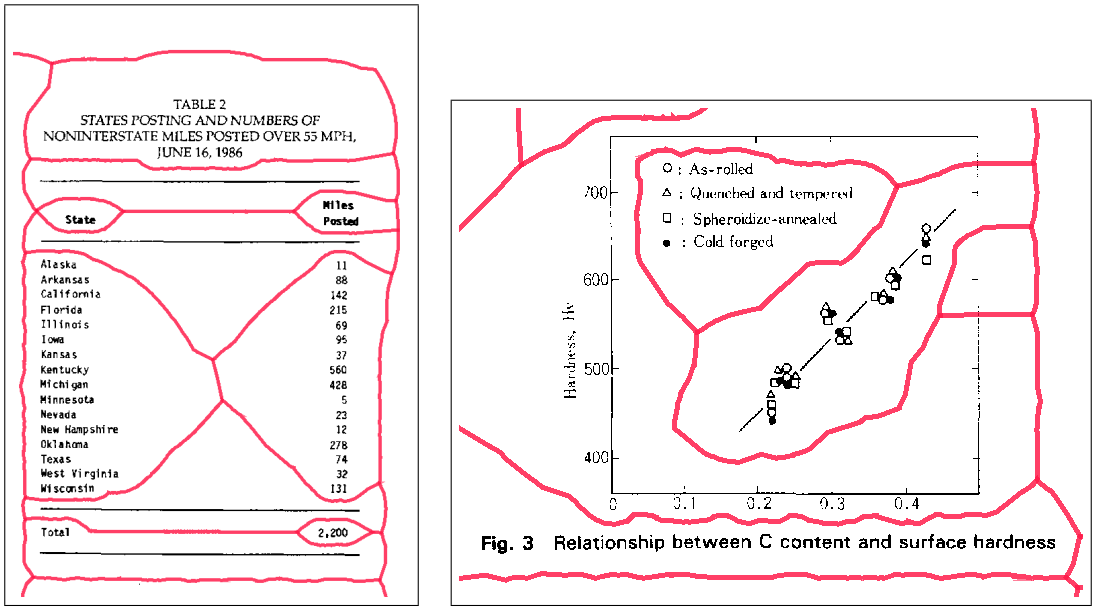
Figure 3.29: Faulty segmentation on noisy documents.

(a) Overlapping Zones

(b) Nested Zones

(c) Merged Zones

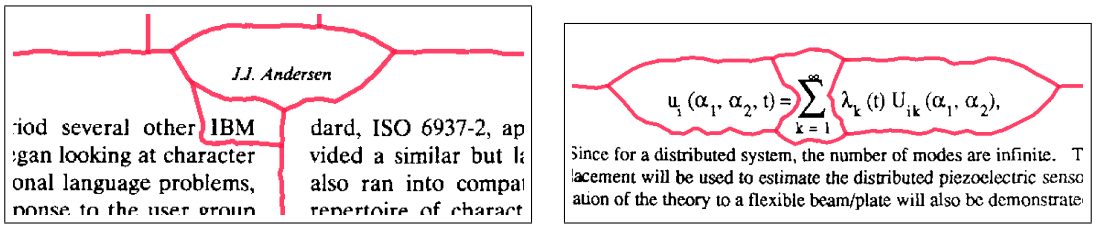
Figure 3.30: (a), (b) Ambiguous and (c) incorrect ground-truthing for component-based segmentation.



(a) (b)

Figure 3.31: Incorrect segmentation for Tables and Charts. (a) Voronoi++ groups similar-looking parts, but tends to group spatially separated columns into zones. (b) Voronoi++ groups similar-looking parts, but tends to separate highly dissimilar parts of an image from the rest.

Abbreviations and Punctuations: Voronoi++ is, sometimes, over-sensitive in distinguishing abbreviations and punctuations from its neighbors based on their dissimilar size and pattern (Figure 3.32a).

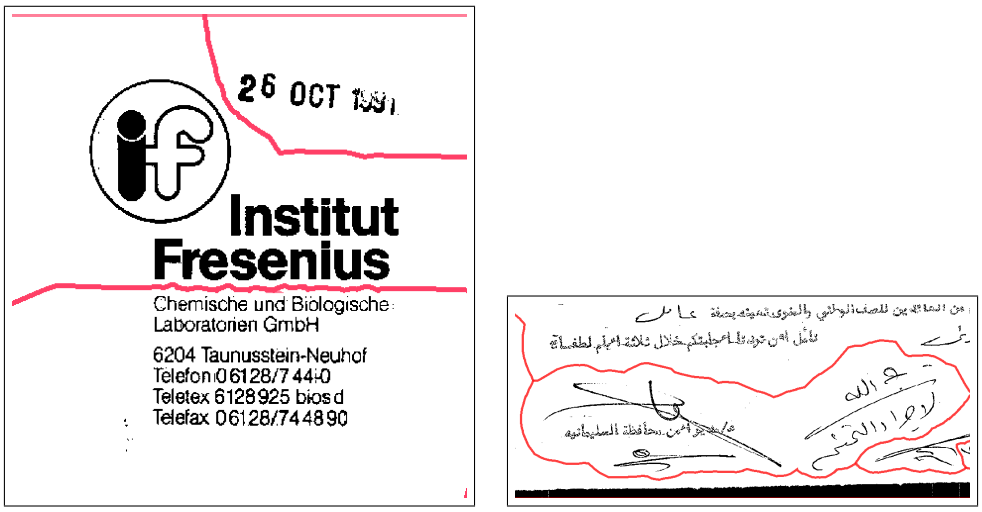


(a) (b)

Figure 3.32: (a) Smaller-sized punctuations around capitalized abbreviations tend to make Voronoi++ over-sensitive for certain regions. (b) Over-segmentation of Mathematical Equations.

Mathematical Equations: Equations are composed of various dissimilar symbols which are often separated by their sizes and distances as compared to surrounding text (Figure 3.32b).

Separation and Similarity context: Separation and similarity contexts in Voronoi++ sometimes fail to understand the nuances in separate regions as illustrated in Figure 3.33



(a) (b)

Figure 3.33: (a) Separation context fails due to curved component (b) Similarity context treats the merged zones similar.

3.3 Zone Classification

After page segmentation, we obtain a set of zones from a document image. The next goal is to classify the segmented zones, in order to allow the application of content-specific algorithms such as Optical Character Recognition (OCR) for textual zones. We have developed a new combination scheme of structural and texture features to represent each region for zone classification. We also developed a novel hybrid classification approach instead of classic one-against-all or one-against-one approaches to enhance accuracy.

3.3.1 Feature Extraction

The proposed page segmentation method is used to segment the document image into a number of zones. Document zones are regarded as image regions with similar structure and texture. Therefore, both structural and textural feature extraction techniques are employed. *Structural features* are run-length based features derived from foreground and background pixels separately. These signature-like features encode pixel distribution information in a given zone [93]. *Textural features*, on the other hand, are based on spatial distribution of all the pixels (spatial structure) and the amount of local image texture (contrast) [66]. In the following, {#f/F} denotes the number of extracted features (f) by cumulative features (F).

3.3.1.1 Structural Features

Run Length Features: For each zone, run-length features are computed for each line along four different directions: horizontal (h), vertical (v), left-diagonal (l) and right-diagonal (r) and are computed for both foreground (0) and background (1) pixels, as follows [93]:

A run-length $\mathfrak{R}$ is the number of contiguous foreground or background pixels in a given direction. Eight bins of run-lengths are created in four directions for foreground and background pixels. For each bin, the following information is extracted.

1. Number of run-lengths: {#8/8} $|\mathfrak{R}_d^p|$

2. Run-length mean features: {#8/16}

$$rlMean_d^p = \frac{1}{|\mathfrak{R}_d^p|} \sum_{rl \in \mathfrak{R}_d^p} rl \tag{3.21}$$

3. Run-length variance features: {#8/24}

$$rlVar_d^p = \frac{\sum_{rl \in \mathfrak{R}_d^p} rl^2}{|\mathfrak{R}_d^p|} - (rlMean_d^p)^2 \tag{3.22}$$

where $d \in \{h, v, l, r\}$ and $p \in \{0, 1\}$. $\mathfrak{R}_d^p$ denotes run-length bin in d^{th} direction for p (=0 for foreground, =1 for background) pixel.

Autocorrelation features {#32/56}: For each direction d, a scan from one end of zone to another, constitutes a pass ρ_d. Four functions for each direction $d = h, v, r, l$ are defined based on passes [93].

1. *Function of pass projections.* Sum of all foreground run-lengths in a pass is called pass projection. $proj_d = \sum_{rl \in \rho_d} rl$

2. *Function of number of pass run-lengths.* For each pass, the number of elements (run-lengths): $|\rho_d|$

3. *Function of pass means.* Mean on each pass: $pMean_d = \frac{proj_d}{|\rho_d|}$

4. *Function of pass spatial means.* For each run-length (rl) in a pass, *position* and *length* parameters:

   $pos_{h,rl} = x_{h,s}, \qquad leng_{h,rl} = x_{h,e} - x_{h,s},$

   $pos_{v,rl} = y_{v,s}, \qquad leng_{v,rl} = y_{v,e} - y_{v,s},$

   $pos_{l,rl} = x_{l,s}, \qquad leng_{l,rl} = x_{l,e} - x_{l,s},$

   $pos_{r,rl} = x_{r,s}, \qquad leng_{r,rl} = x_{r,e} - x_{r,s},$

   where for a direction d, $(x_{d,s}, y_{d,s})$ and $(x_{d,e}, y_{d,e})$ denote the *start* and *end* of a run-length respectively.

   The spatial mean for each pass is:

$$pSPMean_d = \frac{1}{proj_d}\Big(\sum_{rl \in \rho_d} pos_{d,rl} * leng_{d,rl} + \frac{1}{2}\Big(\sum_{rl \in \rho_d} (leng_{d,rl})^2 - proj_d\Big)\Big)$$

Two autocorrelation features are computed for each function [93]. The first feature is the index where the autocorrelation function goes to 10% of its maximum value and second is the slope of the function at indices close to zero.

Foreground Features {#1/57}: The fraction of foreground pixels to the total number of pixels in a given zone is also computed. This feature reflects the density of

pixels in the zone which generally decreases from half-tones to text to diagrams.

3.3.1.2 Texture Features

Texture features compliment structural features because they capture the spatial (local and global) distributions of zone pixels.

Global Spatial Features {#8/65}: Spatial features are low order statistics that are used to capture the foreground pixel distribution information [93]. Spatial features are defined as

$$\begin{aligned} spMean_d &= \frac{1}{N} \sum_{proj_d \in \xi_d} w_d * proj_d \\ spVar_d &= \frac{1}{N} \sum_{proj_d \in \xi_d} [proj_d * (w_d - spMean_d)^2] \end{aligned} \tag{3.23}$$

where N is the number of foreground pixels, ξ_d is the set of pass projections and w_d is a weight that depends on the location of a pass within a zone, for each direction d. Here we use $w_h = y_1$, $w_v = x_1$, $w_l = x_1 + y_1$ and $w_r = y_2 - x_2$, where (x_1, y_1) and (x_2, y_2) define the start and end of a pass respectively.

Local Binary Pattern Features {#256/321}: Local Binary Pattern (LBP) features are an excellent measure for local distribution of binary textures [66]. They capture rotation and gray-scale invariant texture properties. A texture T in a local neighborhood of a binary image is defined as a joint distribution of gray levels of $P > 1$ image pixels, as shown in Equation 3.24

$$T = t(g_c, g_0, ..., g_{P-1}) \tag{3.24}$$

where gray value g_c is of the center pixel of local neighborhood and $g_p(p = 0, ..., P-1)$ corresponds to the gray values of P equally spaced pixels on a circle of radius $R > 0$ that forms a circularly symmetric neighbor set [66]. LBP exhibit the following properties.

Gray-scale Invariance: The gray-value of central pixel is subtracted from neighborhood pixels giving $T = t(g_c, g_0 - g_c, ..., g_{P-1} - g_c)$. Assuming $g_p - g_c$ are independent of g_c, factorization results in $T \approx t(g_c)t(g_0 - g_c, ..., g_{P-1} - g_c)$, where expression $t(g_c)$ describes the overall luminance of the image. In order to achieve invariance with respect to gray-scale shifts in joint difference distribution, only signs of the differences are considered, as follows

$$\begin{aligned} T &\approx t(s(g_0 - g_c), ..., s(g_{P-1} - g_c)) \\ \text{where } s(x) &= \begin{cases} 1 & x \geq 0 \\ 0 & x < 0 \end{cases} \end{aligned} \tag{3.25}$$

LBP is then defined by a number as

$$LBP_{P,R} = \sum_{p=0}^{P-1} s(g_p - g_c)2^p \tag{3.26}$$

and it is invariant against any monotonic transformation of gray scale.

Rotation Invariance: In order to attain rotation invariance, a unique identifier is assigned to each LBP as follows

$$\begin{aligned} LBP^{ri}_{P,R} &= min\{ROR(LBP_{P,R}, k)\} \\ \text{and } k &= 0, \ldots, P-1 \end{aligned} \tag{3.27}$$

where $ROR(x, k)$ performs a circular right-shift on the P-bit number x, k times. It is observed that over 90% of the texture samples have 0, 1 or 2 bit-transitions (0/1) in

the neighborhood and hence inclusion of noisy estimates (with bit transitions < 2) in the dissimilarity analysis degrades performance. Hence, LBP feature operator is modified as:

$$LBP^{riu2}_{P,R} = \begin{cases} \sum_{p=0}^{P-1} s(g_p - g_c) & U(LBP_{P,R}) \leq 2 \\ P+1 & \text{otherwise} \end{cases} \quad \text{where } U(LBP_{P,R}) \text{ captures bit-transitions.}$$

Moreover, a rotation invariant measure of local variance is used, as defined by the Equation

$$VAR_{P,R} = \frac{1}{P}\sum_{p=0}^{P-1}(g_p - \mu)^2 \ , \ \mu = \frac{1}{P}\sum_{p=0}^{P-1} g_p \tag{3.28}$$

Both $LBP_{P,R}$ and $VAR_{P,R}$ provide a very powerful rotation invariant measure of local image texture.

3.3.2 Classification

3.3.2.1 Dimensionality Reduction using PLS

Partial Least Squares (PLS) is a statistical technique that was first introduced in econometrics [95] and later used in other computational fields. It combines the strengths of Principal Component Analysis (PCA) and Multiple Regression (MR). Briefly, PLS aims to analyze the relationship between a set of independent D-dimensional $\mathcal{X} \subset \mathcal{R}^D$ variables (i.e. observations of features) and a set of dependent variables $\mathcal{Y} \subset \mathcal{R}^C$, where C is the dimensionality of the dependent variable space. As a result, PLS estimates a set of orthogonal latent variables which can best relate dependent to independent variables. For a detailed discussion of PLS, the reader is refered to [95].

We use PLS for two purposes – reducing the dimensionality of the problem and improving the classification accuracy. Let $\mathcal{Y} = \{a,\ b\}$ be the set of class labels for classes c_a and c_b. Given a set of vectors $\mathcal{X}$ from zone types c_a and c_b and their associated class labels, we apply PLS to obtain the latent structure $\mathbf{W}_{a,b}$ relating $\mathcal{X}$ and $\mathcal{Y}$. The latent structure $\mathbf{W}_{a,b}$ is a $D \times K$ matrix, where K is the dimensionality of the latent space and $K << D$. Hence, a K-dimensional feature vector $\hat{\mathcal{X}}_{a,b}$ can be estimated as shown in Equation 3.29.

$$\hat{\mathcal{X}}_{a,b} = \mathbf{W}'_{a,b}\mathcal{X} \tag{3.29}$$

$\hat{\mathcal{X}}_{a,b}$ is then the projection of $\mathcal{X}$ onto the latent structure relating the zone pair $\{c_a, c_b\}$ to their corresponding class labels. The dimensionality of $\hat{\mathcal{X}}_{a,b}$ is much smaller than that of $\mathcal{X}$. Furthermore, $\hat{\mathcal{X}}_{a,b}$ better discriminates between c_a and c_b because PLS maximizes the covariance between $\mathcal{X}$ and $\mathcal{Y}$ variables.

3.3.2.2 Hybrid Multi-Class Classification

The features extracted as discussed in Section 3.3.1 are combined to create a D-dimensional feature vector $\mathcal{X}$. The zone identification problem can be formulated as follows: *given a D-dimensional feature vector $\mathcal{X} \subset \mathcal{R}^D$, classify $\mathcal{X}$ into a set of C zones type, where $C > 2$, based on the zone content*. [1]

Classically, this multi-class problem has been treated by constructing either C one-against-all binary classifiers or $\frac{C\ (C-1)}{2}$ one-against-one binary classifiers and then using a voting scheme to obtain the required class label, with the latter method reported to be the most successful one. However, one-against-one

[1] Note that if $C = 2$, the problem then is a straightforward binary classification one.

methods suffer a principal limitation. If the observation being tested does not belong to either of the two classes on which the classifier is trained, a vote will be incorrectly cast, biasing the final classification outcome. In our proposed method, we use a hybrid of two approaches. Let

$$y_{a,b} = f_{a,b}(\mathcal{X}) \tag{3.30}$$

be a binary classifier that maps an input vector $\mathcal{X}$ into one of the two classes c_a or c_b. For all pairs of arbitrary zone classes $\{c_i,\ c_j\}$, where $i, j = 1, \ldots, C$ and $i \neq j$, we construct two binary classifiers, as shown in Equation 3.31

$$\begin{aligned} y_{i,j} &= f_{i,j}(\hat{\mathcal{X}}_{i,j}) \\ y_{i-j,all} &= f_{i-j,all}(\hat{\mathcal{X}}_{i-j,all}) \end{aligned} \tag{3.31}$$

where $\hat{\mathcal{X}}_{i,j}$ is the projection of training vector $\mathcal{X}$ onto the latent structure relating $\{c_i, c_j\}$ to the corresponding class labels. $\hat{\mathcal{X}}_{i-j,all}$, on the other hand, is the projection of $\mathcal{X}$ onto the latent structure relating a combination of c_i and c_j from one side and all other classes from the other, and their corresponding class labels. Hence, $f_{i-j,all}$ is two-against-all classifier, which we call the *indicator classifier*. In order to solve the multi-class classification problem, we construct $\frac{C\ (C-1)}{2}$ one-against-one classifiers plus $\frac{C\ (C-1)}{2}$ indicator classifiers.

At testing time, to determine the zone type of an input vector $\mathcal{X}$, all $\frac{C\ (C-1)}{2}$ pairs of classifier must be used. To check if an input vector $\mathcal{X}$ belongs to c_i or c_j, the vector is first projected onto $\hat{\mathcal{X}}_{i-j,all}$. Subsequently, if $f_{i-j,all}$ *indicates* that $\mathcal{X}$ does not belong to either c_i or c_j, then $f_{i,j}$ is not used and no vote is being cast. Otherwise, $\mathcal{X}$ is projected onto $\hat{\mathcal{X}}_{i,j}$ and $f_{i,j}$ is used to cast the vote accordingly. The

class with the maximum number of votes is selected to be zone type of $\mathcal{X}$.

3.3.3 Evaluation

The proposed method was applied to the University of Washington (UW) dataset. The dataset contains 1690 document images with a total number of 24531 zones. We consider 10 different zones types – chemical drawing, small text and symbols, drawing, halftone, logo or seal, map, math, ruling, table and large text. Using SVM as the underlying binary classifier, the hybrid classifier achieves 97.3% classification accuracy. To our knowledge, the best reported performance on this dataset is 98.45% of Wang et al. [93]. However, the UW dataset is significantly unbalanced with 87.9% small text samples, 0.065% logo and seal samples and 0.057% map samples.

In order to further assess our proposed algorithm, we eliminated small text, logo and seal and map classes from the dataset. The hybrid classifier achieves a comparable 96.6% accuracy on the remaining 7 classes. No result is available for a similar experiment from [93]. Moreover, similar experiments show that the hybrid classification scheme out-performs the classic one-against-one scheme. Table 3.4 summarizes the results.

	1-vs-1	Wang et al. [93]	Hybrid
Un-balanced	93.1%	98.45%	97.3%
Balanced	88.2%	N/A	96.6%

Table 3.4: Performance Comparison

Chapter 4

Character Segmentation and Recognition

4.1 Background and State-of-the-art

Current trainable OCR systems typically assume

1. the availability of ground-truth data at character level for training
2. a sufficient number of representative samples of each class and/or
3. the accurate character segmentation is available before recognition

Low-density languages and languages without commercial OCR engines challenge these assumptions in various ways. In languages with large character sets, the overall representation of classes typically decreases in typical texts with some classes having no representation unless a very large amount of training data is provided. Such data may be difficult to obtain, and not practical for such languages. Another problem for syllabic languages is that the nature of their script can not ensure character-level data availability for training and one-dimensional character segmentation routines fail for two dimensional positioning of glyphs inside syllables. Broken and touching characters add to the woes and require a more intelligent character segmentation methodology. With this, a generic character recognition system suggests the need for a script-independent specialized character segmentation routine.

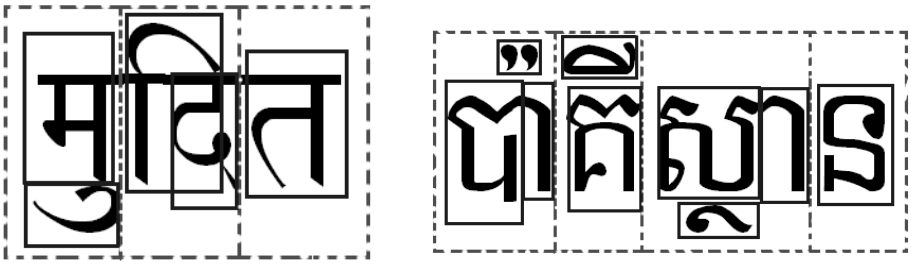
Most systems focus on feature extraction and classification to improve accuracy but they require training and the availability of class samples at the character and/or word levels. The objective of our work is to create a generic script recognizer which can be bootstrapped from font descriptors and can be trained using a minimal number of samples. Since our research is targeted towards low-density languages, the availability of large amounts of ground-truth data can not be assumed [55]. For this reason, many techniques such as SVMs and HMMs which require large amounts of training data, can not be used. In addition, limited user feedback is a key to the system's adaptiveness.

4.2 Base Recognition System

The contributions of this work stem from observations about a base system we have in [55, 54]. This section describes the system which contains three different functional components: (1) Hierarchical segmentation (2) Feature Extraction and (3) Classification

4.2.1 Segmentation

It is often non-trivial to segment characters for complex non-Latin (especially syllabic) scripts (Figure 4.1a,b), and availability of segmented characters can not typically be assumed for training purposes. Hence, Maryland researchers developed a limited user feedback mechanism in which unregistered electronic text is fed to the training module along with the document image [55]. Text-



(a) (b)

Figure 4.1: Bigger boxes (dotted) demonstrate syllable-level segmentation while smaller (solid) ones show character-level segmentation for (a) Devanagari script (b) Cambodian script composition.

alignment is accomplished by aligning zones, lines and words in image and electronic document text in a hierarchical manner. Since the alignment can be imperfect, only the best-match segments are returned.

In [54] syllabic and non-syllabic scripts require different procedures for character alignment and segmentation. The user must specify script category (syllabic or non-syllabic) and its parameters.

The system follows a dissection based character segmentation approach and each connected component in a word is associated with accents or separate dots above or below it, to form a glyph. With the assumption that a character won't be too wide or too narrow, a glyph satisfying the following conditions is considered a character if:

1. The aspect ratio falls in the range $[r_{low}, r_{high}]$;

2. The area is larger than A_{min};

where r_{low}, r_{high} are the predefined low and high aspect ratio thresholds respectively and A_{min} is the area threshold (derived from the data, the values were found

to be 0.2, 1.0 and 5 respectively). Character images are extracted by aligning characters in the ground-truth word-text and document word-image.

4.2.2 Feature Extraction

After character segmentation, each character is processed through a feature extraction routine where the most descriptive or differentiating features are extracted and used in training and testing. Feature extraction [37, 90] and classification [29, 71] form the basis of any training and testing process. Feature extraction approaches fall into two classes - spatial domain [31] and transform domain [38, 56] based. Spatial domain approaches derive features directly from the pixel representation of the pattern. With a transform domain technique, the pattern image is first transformed into another space using, for example, Fourier, Cosine, Slant or Wavelet transform and features are derived from the transformed images [38].

In our system, three feature extraction routines have been developed and can be used interchangeably:

1. *Template initialization:* Each character image is first resized to a 32 X 32 vector map. A probabilistic template is generated from all samples of each class from the training data [55]

2. *Zernike Moments:* Moment descriptors have been studied for image recognition and computer vision since 1960s. Teague [91] first introduced the use of Zernike moments to overcome the shortcomings of information redun-

dancy present in the popular geometric moments. Zernike moments are a class of orthogonal moments which are rotation invariant and can be easily constructed to an arbitrary order. Khotanzad & Hong [45] showed that Zernike moments are effective for the optical character recognition (OCR).

3. *Directional Features:* Templates are rigid, and can result in poor models for classification for noisy documents. Zernike moments, however, are a transform-based feature analysis method and more robust to shape variances, but do not utilize inherit 'directional' property of complex scripts. The relative placement of neighboring pixels is more important than the overall placement of pixels forming the character. Directional features [44] record local pixel positions for each contour pixel and generate a feature vector using that information.

   For directional features, the character image is normalized, and the contour is extracted and mapped to a 64-by-64 mesh. The mesh is divided into 49 (7-by-7) sub-areas of 16-by-16 pixels where each sub-area overlaps eight pixels of adjacent sub-area (Figure 4.2). For each sub-area, a four-dimensional vector (x1, x2, x3, x4) is defined where x1, x2, x3 and x4 record the relative direction (vertical, horizontal, forward inclined, backward inclined) of neighboring pixels with respect to each foreground pixel in the sub-area. Hence, a 49 x 4 = 196 unit long feature vector is produced. Figure 4.2 shows the directional feature extraction process step-by-step.

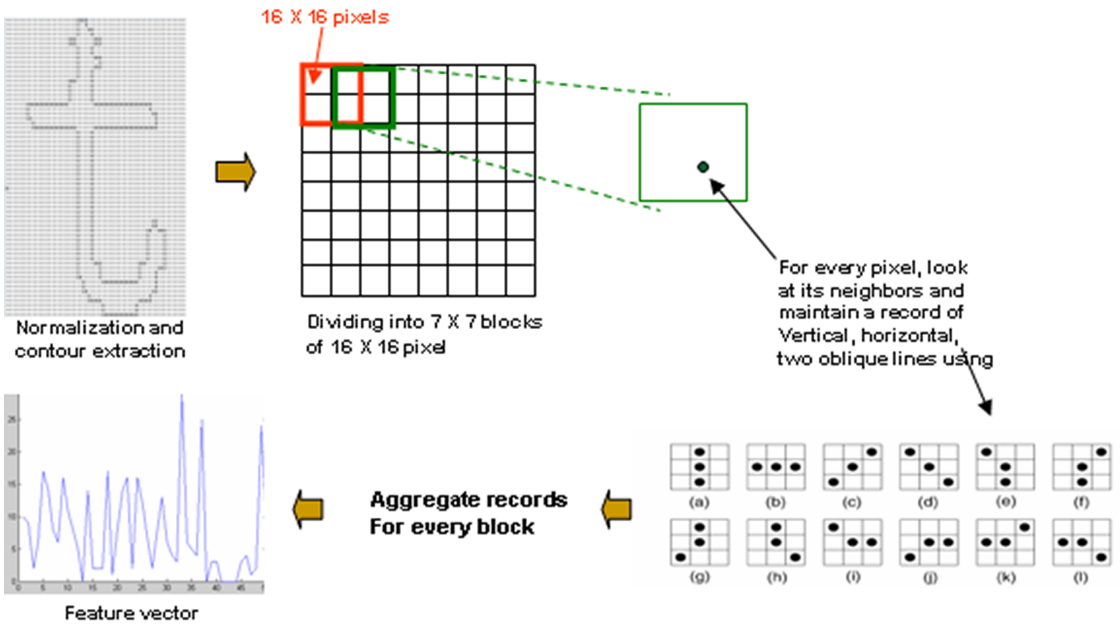


Figure 4.2: Directional Element Feature Extraction.

4.2.3 Classification

Classifiers like Artificial Neural Network [17] or Support Vector Machines (SVMs) [7] have been successful at recognizing various non-Latin scripts. These can work on either spatial or transform based features. Hidden Markov Models (HMMs) [21, 72] work on a large number of training samples to estimate probability parameters. They have been quite successful in handwriting and speech recognition. Fuzzy rules [27], Mahalanobis and Hausdorff distance, and Evolutionary algorithms [86] are other techniques used for recognition.

We have three different classifier modules in our recognition system, which can be called interchangeably based on the input parameters, and are known to do fairly well with limited training data.

4.2.3.1 Template Matching

Awarding probabilities when template pixel matches with the corresponding pixel in a candidate character image and penalizing otherwise, forms the core objective of template matching. The template which has the best match is selected as the class. The candidate character image is binary, while the pixel values of the template map g(x, y) are in a range [0,N_{inst}]. The similarity of a character image f(x,y) and a template $g_b(x, y)$ is defined by a weighted similarity.

$$S_w(f, g) = 1.0 - \frac{1}{N^2} \sum_{x=1}^{N} \sum_{y=1}^{N} \omega(x, y) \left| f(x, y) - g_b(x, y) \right|$$

where the weight $\omega(x, y)$ is defined as:

$$\omega(x, y) = \begin{cases} 1.0 & \text{if } g_b(x, y) \text{ is background} \\ \frac{g(x,y)}{N_{inst}} & \text{if } g_b(x, y) \text{ is foreground} \end{cases}$$

4.2.3.2 Nearest Neighbor Classifier and Weighted Euclidean Distance

Nearest Neighbor Classifier is used on Zernike Moment features with a simple weighted Euclidean distance (WED). For each test sample, the classification is based on the distance between this sample and each class. The feature vector is in a d-dimensional space and the computed mean and standard deviation feature vectors for class i are $\mu^{(i)}$, $\alpha^{(i)}$, where i = 1...M and M is the number of classes. For each test sample $x \epsilon R^d$, the distance between this sample and each class is computed using the following formula:

$$d^{(i)}(x) = \sum_{k=1}^{d} \left| \frac{x_k - \mu_k^{(i)}}{\alpha_k^{(i)}} \right|$$

4.2.3.3 Hierarchical Classification

Kanji and South-East Asian scripts have a large number of symbols. Hence, one-stage discrimination does not generally suffice. In this approach, two-stage classification (coarse and fine) is used. The aim of coarse classification is to cluster similar-looking characters into groups and then perform fine classification to extract the right class [44].

1. *City Block Distance with Deviation (CBDD):* Let $v = (v_1, v_2, ...v_n)$ be an n-dimensional input vector and $\mu = (\mu_1, \mu_2, ...\mu_n)$ be the standard vector of a category. The CBDD is defined as:

$$d_{CBDD}(v) = \sum_{j=1}^{n} max\left\{0, \left|v_j - \mu_j\right| - \theta.s_j\right\}$$

   where s_j denotes the standard deviation of j^{th} element, and θ is a constant.

2. *Asymmetric Mahalanobis Distance:* For each cluster, the correct class is obtained by finding the minimum asymmetric Mahalanobis distance from the templates in that cluster. The function is given by:

$$d_{AMD}(v) = \sum_{j=1}^{n} \frac{1}{\hat{\sigma_j}^2 + b}(v - \hat{\mu}, \phi_j)^2$$

   where b is the bias, $\hat{\mu}$ is the quasi mean vector of the samples of the class m, ϕ_j is the eigenvector of covariance matrix of this category and $\hat{\sigma_j}$ is the quasi variance. In case of a tie, N-nearest neighbor is used, with N = 3.

4.2.4 Additional Challenges

An analysis of results of the system described above shows that a large number of recognition errors are caused by incorrect character segmentation of complex syllabic scripts or by touching and broken characters (in degraded documents). Due to the large glyph set and possible conjunct set of such scripts, limited ground truth could not cover all the possibilities and hence many classes had no representation in the ground truth data.

Some of the complex problems in the field of syllabic character segmentation have already been listed. This implies that the benefits of good feature extraction modules (followed by classifiers or their combinations) can not be realized until there is a robust generic solution to the character segmentation problem.

4.3 Font based Intelligent Character Segmentation

4.3.1 Benefits and Font Models

Nearly every script considered has a representative TrueType font. One feature of a TrueType font file is an explicit, generative model for layout of text. Given a character, the position of next character can be predicted using the properties of these fonts. This will be used to aid in segmenting touching characters, grouping broken characters and processing glyphs fused or overlapped in syllables. Another advantage of such an approach is that it does not entail script dependent mechanisms for segmentation and aims at a generic character segmentation

algorithm for any given script. This method branches off of the second tier of character segmentation approaches (Section 1.3.1.1) by generating well-defined component extraction and segmentation hypotheses.

Font-files have a wealth of information [47] and can be used to produce this generative model. Information in the font-files includes a list of characters, glyphs of each character, font ascenders and font descenders.

At a given font size, the file also contains the following information for each character

- Unicode value
- Height and Width
- Horizontal Advance (HA): the horizontal distance between the origins of present and next character in a word
- Vertical Advance (VA): the vertical distance between the origins of present and next character in a word
- Bounding Box (BB)
- Left Bearing (LB): the horizontal distance between the left-end of a bounding box and its origin
- Right Bearing (RB): the horizontal distance between the right-end of a bounding box and origin
- Combination rules of ligatures

Many parameters are redundant, as they can be derived from other parameters. Font-files for similar fonts can be analyzed for consistency in these model-parameters. Figure 4.3 shows the location of characters for three fonts of Devanagari script (at the same font size). The fonts place each character or sub-character glyph at nearly the same position (with respect to a given origin), hence demonstrating the consistency in these parameters. Similar analysis was done for non-syllabic scripts.

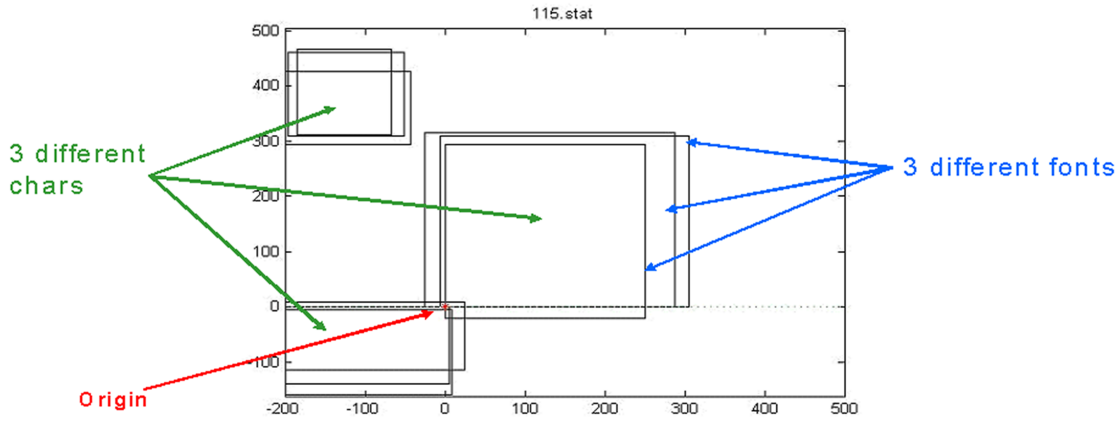


Figure 4.3: Chart shows locations of 3 different characters of Devanagari script using 3 structurally similar Devanagari font files.

For a given font face and size, a word is rendered by placing the first character using its bounding box. Using the horizontal-advance, vertical-advance and origin of present character, the origin of next character is determined. Using this origin, the next character is placed in its bounding box and the process is repeated for the remaining characters in the word. Figure 4.4 shows the process.

Using a group of structurally similar fonts, glyphs can be extracted and used for training purposes. This eliminates the problem of coverage in the large

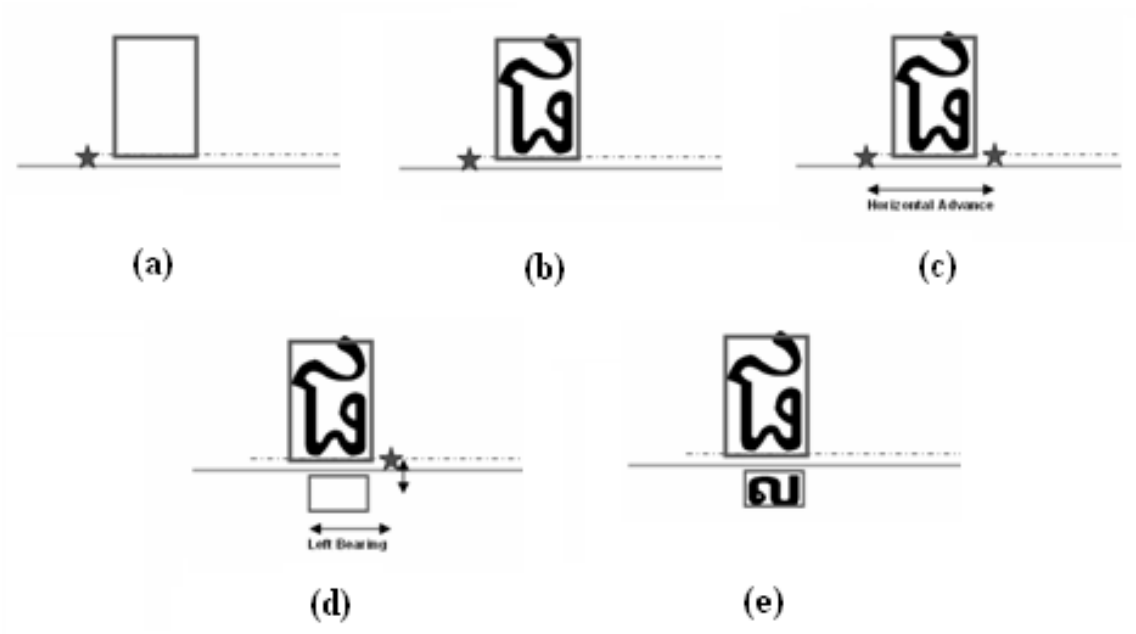


Figure 4.4: The figure above shows rendering of characters in a word in Khmer using font-models (a) Locating first character (b) Placing character into its bounding-box (c) Determination of the origin of next character (d) Determination of next character's bounding-box (e) Placing the second character.

alphabets of syllabic scripts with limited ground-truth. The glyphs extracted can be a substitute for missing or rarer character classes from ground-truth. The information is then used to segment characters from word-images during training and testing using the process illustrated above (Figure 4.4).

4.3.2 Training Using Font-files

The following steps describe the process of training a system with limited ground-truth data (Figure 4.5).

Step 1: A group of similar fonts, resembling the text in documents to be processed, are provided along with electronic text.

Step 2: For each character, the average bounding box, horizontal and vertical-advance values are computed from the font files.

Step 3: The character glyphs from font files are generated as templates and passed through the feature extraction routines.

Step 4: Each document image along with its corresponding unregistered electronic-text ground-truth file, is passed through the segmentation module and a hierarchical structure (containing Page → Zone → Line → Word) is created with word-alignments at it's root.

Step 5: Each word in this structure is further segmented into characters using (a) aligned characters in the corresponding ground-truth, (b) font parameters extracted in step 2, and (c) process explained in Figure 4.4.

Step 6: For each character segmented in the document image, feature extraction is performed.

Step 7: Classes are modeled using the features of glyphs from training set and font files. Hence, limited ground-truth data with some unrepresented glyphs suffices, as those glyphs are processed from the font-files.

4.3.3 Segmentation and Recognition

With the objective of grouping broken characters, segmenting conjuncts and touching characters, the technique of font-model based intelligent character segmentation and recognition was developed. As discussed earlier, it falls in second category of character segmentation with an advantage of reducing the

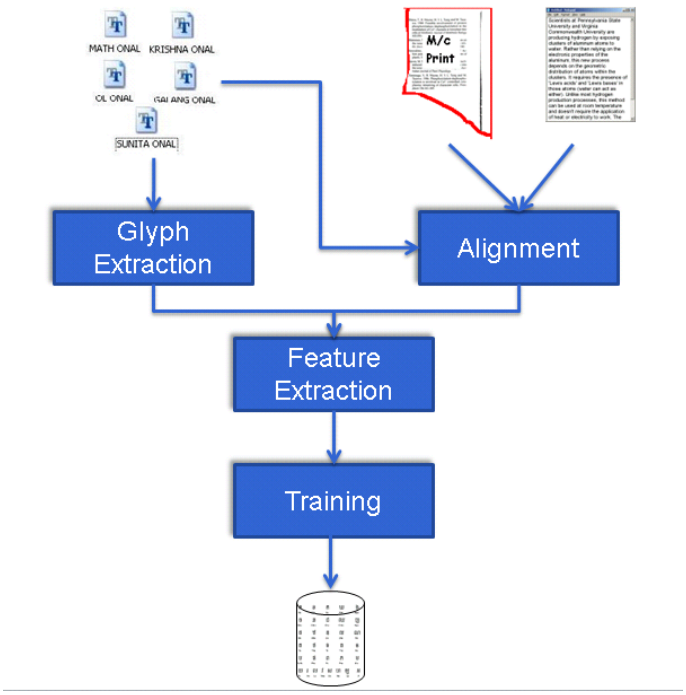


Figure 4.5: Character training using font-files.

number of hypotheses by the knowledge of the next character's position, given the present character. This is achieved using font parameters.

Algorithm: We first define the following:

η_i : Bounding box of possible component combinations,

where $1 \leq i \leq C_1^N + C_2^N + ... + C_N^N$

ρ_j : Bounding box of a predicted character,

where $0 \leq j \leq$ Total number of characters in the language

$\bigcirc_{ij}$: Overlap(η_i, ρ_j)/minArea(η_i, ρ_j)

τ_1 : Threshold on $\bigcirc_{ij}$

$\mathfrak{R}_{ij}$: maxArea(η_i, ρ_j)/minArea(η_i, ρ_j)

τ_2 : Threshold on $\mathfrak{R}_{ij}$

$\gamma : \{\forall \eta_i \; s.t. \bigcirc_{ij} \; > \; \tau_1\}$

$\delta : \{\forall \eta_i \; s.t. \bigcirc_{ij} \; > \; \tau_1 \; \& \; \Re \; > \; \tau_2\}$

The document image is classified into zones, lines and words. For each word, connected component analysis is performed. Assuming a maximum of N uncovered components can be combined together to form a character, there can be $C_1^N + C_2^N + ... + C_N^N$ possible nodes (η_i) for next character (typically N = 3). Given the present character, predictions (ρ_j) are made for the next-character's locations (using font-model). Those η_i which do not overlap (with threshold τ_1) with any ρ_i, are discarded. The η_i which overlap (with threshold τ_1) with any ρ_j are inserted into a set γ. The η_i which enclose any ρ_j are inserted into a conjunct set δ. Nodes of set γ are ranked by their confidences returned from the recognizer. Nodes of the conjunct set δ are given for a conjunct-test (described later). If they pass the test, the conjunct is broken into possible characters using Dijkstra's algorithm and individual character confidences are returned. Only the first character (along with its confidence) from every conjunct is kept in the set δ and later pieces are stacked back into the set of uncovered components. The highest confidence character is picked from set γ and δ combined. The process is repeated for the uncovered connected components in the next stage. In case of dead-ends (when no possible character location coincides with the present connected-component nodes), back-tracking is performed and the path is pruned (Figure 4.6).

Conjunct-test: Conjuncts form an integral part of any syllabic script. Many characters combine to form a single shape. Techniques so far have relied on a

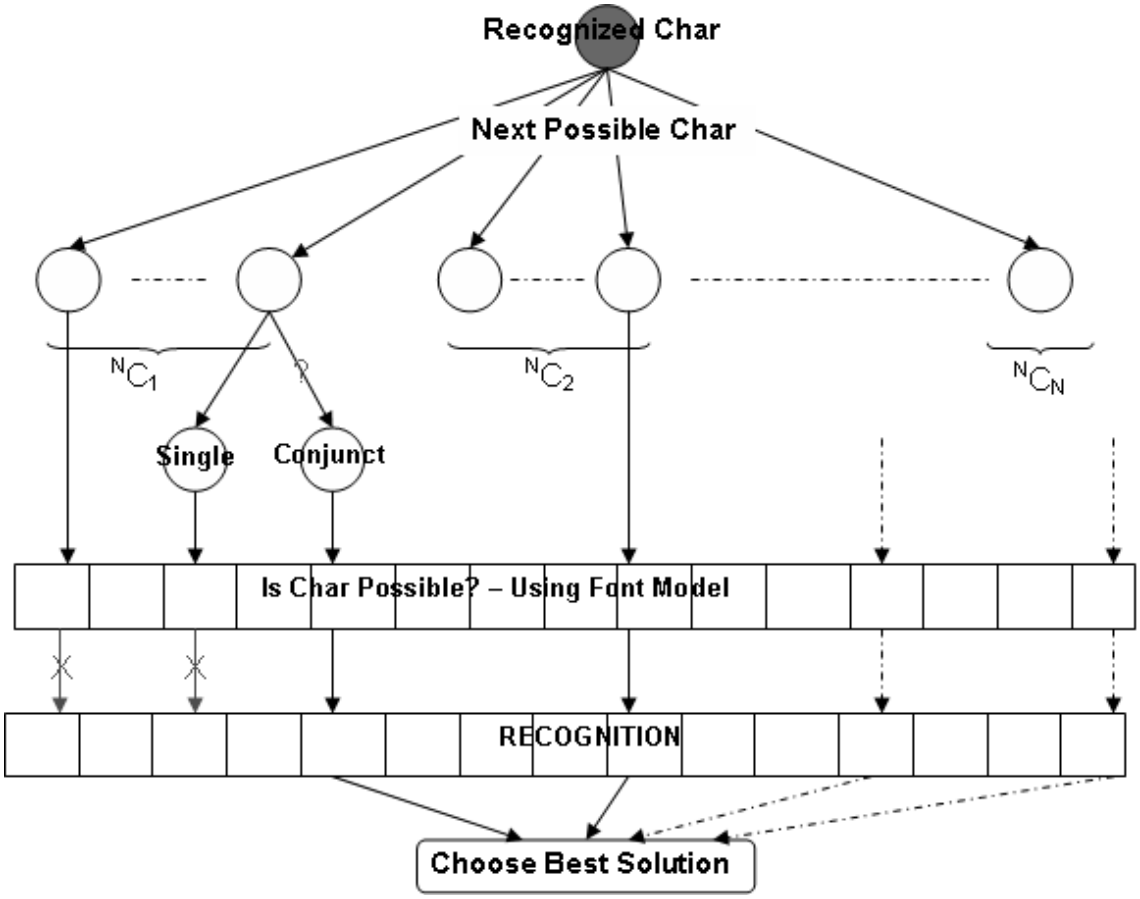


Figure 4.6: Dynamic Network created during best-path search of word-recognition (using Font-models).

crude method of aspect ratio threshold to determine if a character is a conjunct and needs to be broken down further. With font-models, an intelligent conjunct detection procedure has been developed. A glyph is passed for conjunct-analysis only if it encompasses the possibility of two or more characters of the given script under test. This position-analysis can be done only through font-models of a script, as illustrated in Figure 4.7.

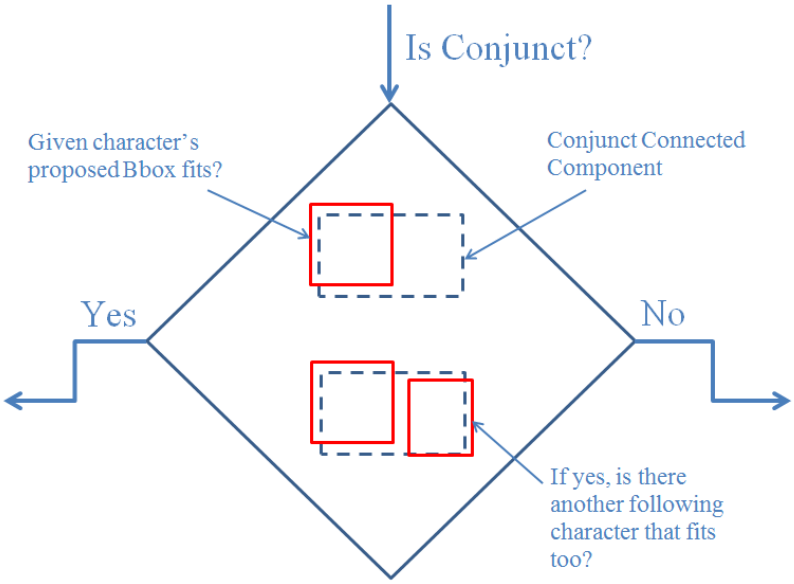


Figure 4.7: Conjunct Detection. Bigger (dotted) Box shows a possible conjunct under detection. If two characters (solid boxes) fit in (using font-model), it may be a conjunct.

4.4 Evaluation

4.4.1 Datasets

Our experiments were performed on two classes of scripts - Latin (non-syllabic) and Khmer (syllabic). Two datasets for English and one for Khmer were used with the following properties:

1. Noisy Latin (D_1): The first Latin dataset has varying amount of clarity across the pages which leads to a large number of broken and touching characters (Figure 4.9). Apart from this, the documents contain noise introduced during the printing and scanning process. The characters in words are also skewed and not aligned perfectly with the word's bottom reference line. This

imposes additional challenges for character segmentation and prediction of next-character position using font-models and verifies robustness of our approach. The closest font is *NSimSum*.

2. Clean Latin (D_2): The second Latin dataset, on the other hand, is a much cleaner dataset, with a font resembling more closely to *Courier New*. A single English document had approximately 2000 characters and 330 words.

3. Khmer (D_3): The Khmer dataset contains some documents from a Cambodian Gazetteer and documents scanned from other sources (15 pages total). These documents are dark and hence suffer badly from touching-characters. This, combined with the presence of numerous conjuncts in Khmer script, becomes an ideal dataset for evaluation of our techniques. The closest font to the documents is of *Limon S1*. A single Khmer document had approximately 1500 characters and 100 words.

Sample text from English and Khmer document is shown in Figure 4.8. In each dataset, three to five documents were chosen for training. The goal was to evaluate our approach with limited user feedback and a limited training set - which is generally the case for any new script under study.

4.4.2 Metrics and Tools

The text returned by the OCR system is matched against the ground-truth data using a tool based on the UNLV Evaluation Toolkit [61]. The evaluation tool prints out an elaborate description of insertion, deletion and substitution errors in

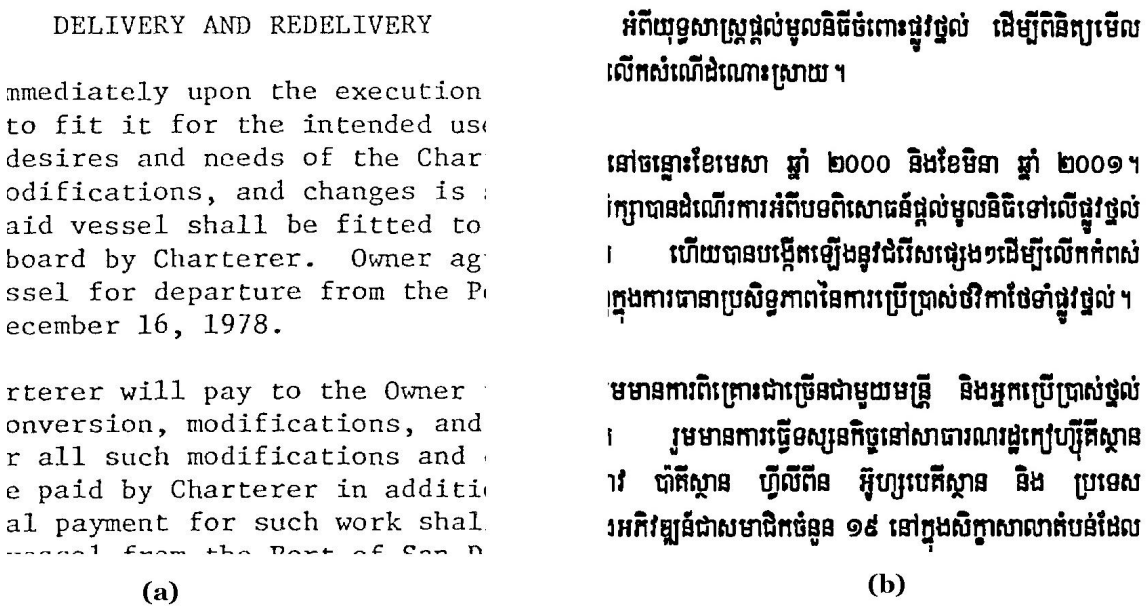


Figure 4.8: (a) Sample text from English dataset (b) Sample text from Khmer dataset.

the form of one-to-one, one-to-two, two-to-one or two-to-two confusions. It also summarizes the most confused characters along with their confusions. Apart from character confusions, it also lists word-confusion matrices in a similar fashion.

4.4.3 Experiments and Results

4.4.3.1 Feature Extraction

Table 1 compares template matching and directional element feature (DEF) extraction results, both for Latin and Khmer documents. A weighted similarity measure (Section 2) was used to classify templates and CBDD was used to classify the directional feature set.

Table 4.1: Compares character level accuracy results for Latin and Khmer scripts using Template and Directional Element Features (DEF).

	English		Khmer	
Accuracy	Template Matching	DEF	Template Matching	DEF
Char	86%	93%	84%	89%

4.4.3.2 Character Segmentation

Quite a few of recognition errors in dissection based character segmentation were due to bad character segmentation. The use of font-models reduced those errors. Extraction of characters from fused ligatures is still a problem and is considered for our future work. Figure 4.9 shows the improvements in character segmentation for both broken and touching characters.

4.4.3.3 Character and Word Recognition

Table 2 summarizes the improvements gained using our font-model based character segmentation and recognition - for both English and Khmer datasets. Character accuracies as well as word accuracies have been reported. The accuracies reported are using directional feature extraction and the CBDD classifier. wo/FM stands for WithOut Font Model (following dissection-based segmentation) and w/FM stands for With Font Model. Due to a much higher word-length in Khmer, the word-accuracies are low as compared to English. Figure 4.10 compares character recognition for both English and Khmer scripts using the

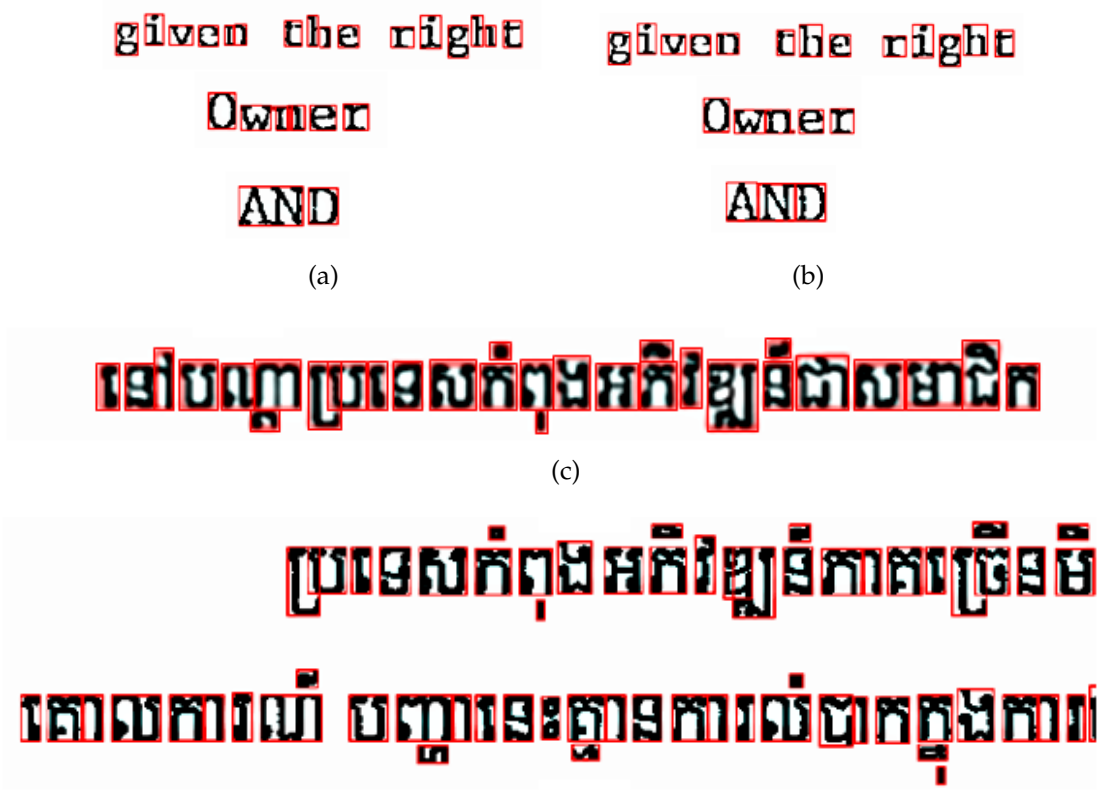


Figure 4.9: Improvements of our technique over older dissection based techniques (a) and (b) show Latin script character segmentation using dissection font-model based techniques respectively (c) and (d) show results for Khmer script using dissection and our font-model based techniques respectively.

developed feature-extraction systems and font-models.

Table 4.2: Compares dissection and font-model based techniques.

	English		Khmer	
Accuracy	wo/FM	w/FM	wo/FM	w/FM
Char	93%	96%	89%	92%
Word	83%	89%	38%	37%

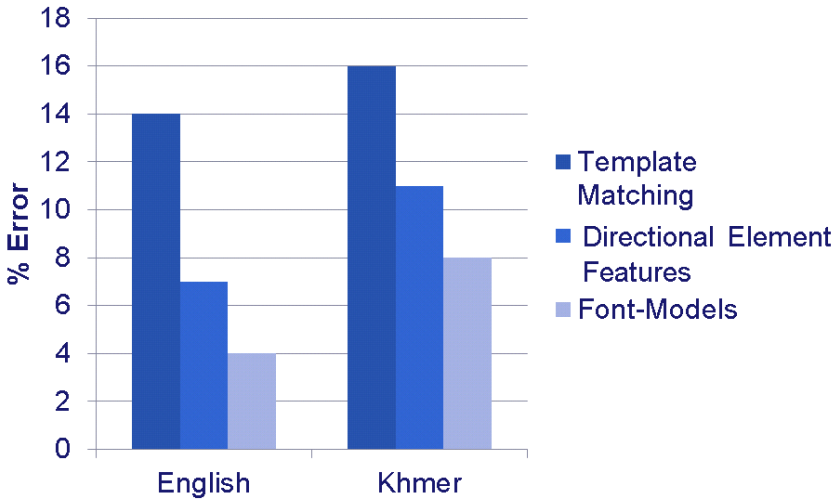


Figure 4.10: Character recognition error rates continue to drop using better feature extraction methods and an intelligent character segmentation system using font-files.

Chapter 5

Summary and Future Directions

5.1 Noise Detection and Removal

Summary: In Chapter 2, we have presented a novel approach toward clutter detection and removal for complex binary documents. Our distance transform based approach aimed at the removal of irregular and non-periodic clutter noise from binary document images and is independent of clutter's position, size, shape and connectivity with text. The novelty of this approach is in its restrictive nature to remove clutter, as text attached to the clutter is neither degraded nor deleted in the process. Residual image creation, as an intermediate step, helps in detecting clutter and determining the clutter-content boundary precisely. Clutter detection and removal accuracies were reported greater than 95% on machine-printed and handwritten documents of English and Arabic scripts.

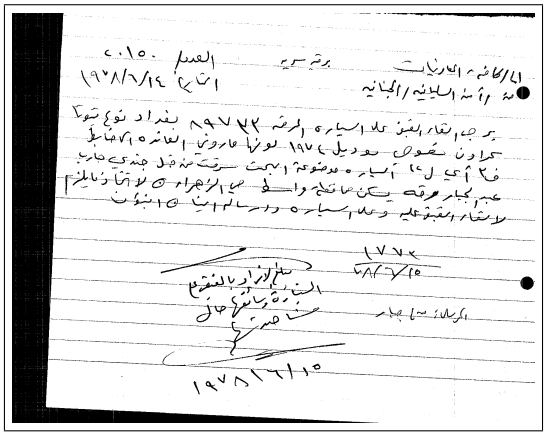
We also presented a novel approach to stroke-like pattern noise (SPN) detection and removal for binary document images. Our two-phased approach aimed at understanding the script-independent prominent text component features as the first step in a supervised classification approach. SVM with an RBF kernel was used to classify these components from the rest using a minimal set of training samples. Later, based on the cohesiveness and stroke-width features of these components, smaller text components are filtered out using k-means

clustering. The novelty of this approach is that it does not aim at script or character recognition in order to perform text extraction at diacritic level. It also does not depend on a sufficient number of representative ground-truth samples at component level for training. Instead, it uses generic script features to divide-and-conquer components into prominent and dependent ones to achieve noise removal. SPN (Stroke-like Pattern Noise) removal was tested on a set of Arabic machine-printed and handwritten documents and precision and recall rates of 98% and 97% respectively were reported.

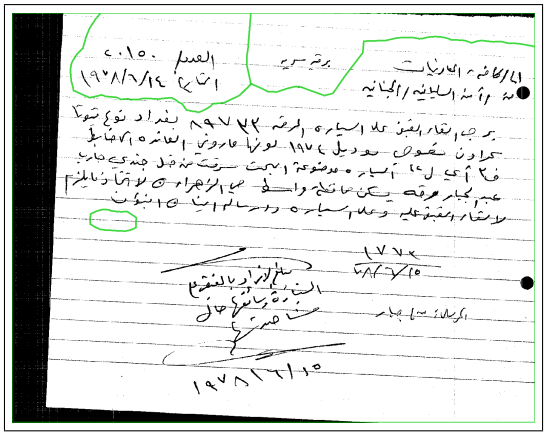
Figure 5.1 shows image enhancement results after clutter and SPN removal, and illustrates their respective Voronoi++ segmentations. Clearly, a cleaner document image has a better chance to succeed in the later stages of the document processing pipeline.

Future Directions: This work can be extended in the following ways:

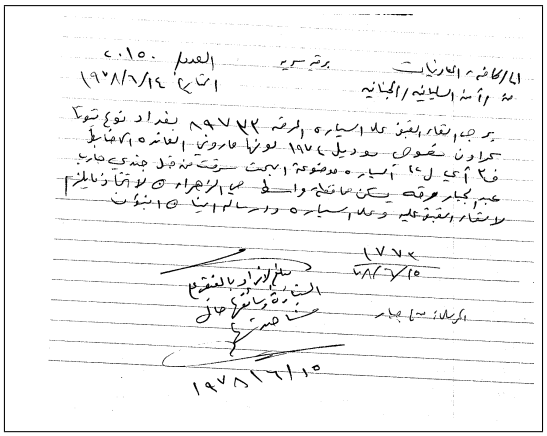
1. **Text recognition based clutter removal:** Section 2.2.1.2 showed that clutter may not always form contrasting boundaries with the background. While avoiding the removal of text, we may also leave some boundary pixels of the noise in the process (as shown in Figure 5.2a). This is because its not always necessary that any thin protrusion from the clutter is text. Clutter inherently may have protrusions from it's main body. Hence, such prevailing pixels after clutter removal may give rise to a new form of noise which may be more difficult to remove. It is important that text identification should be performed before preservation of a protruded branch.



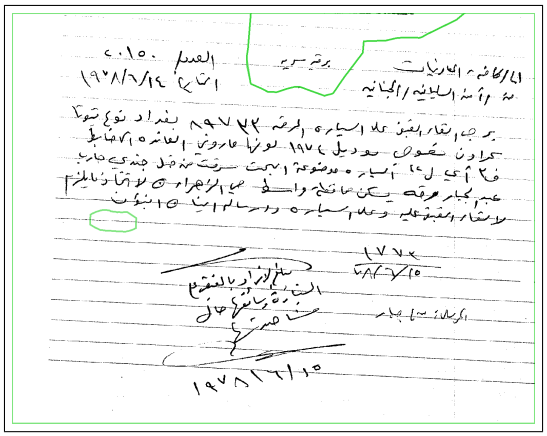
(a) Noisy image



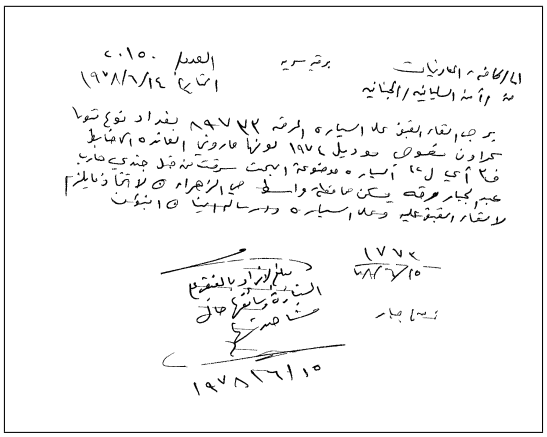
(b) Voronoi++ segmentation on noisy image



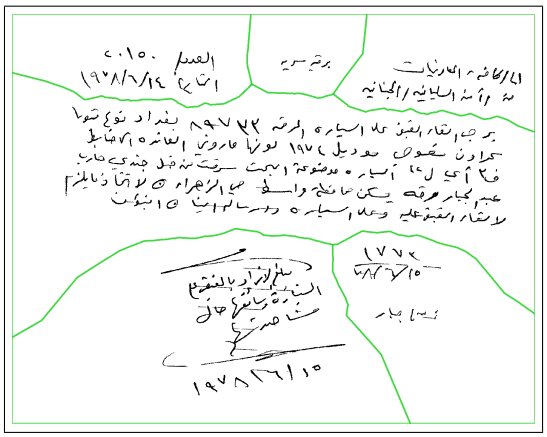
(c) Image with clutter removed



(d) Voronoi++ segmentation on image with clutter removed

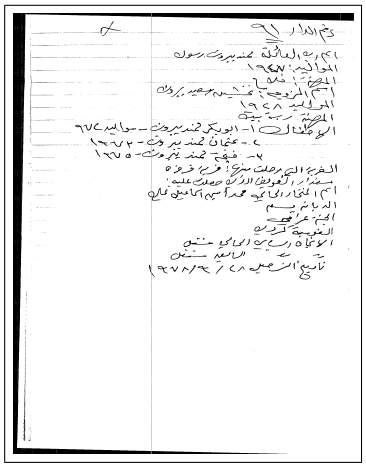


(e) Clean image with SPN removed

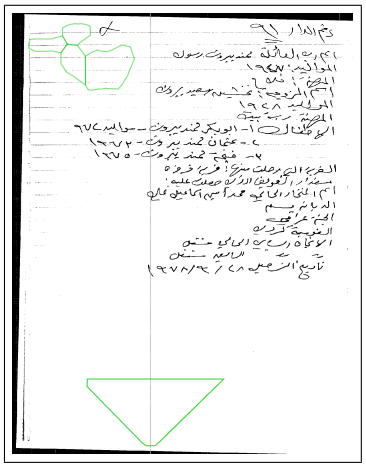


(f) Voronoi++ segmentation on a clean image

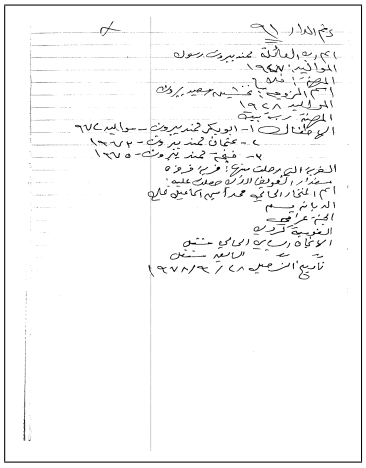
Figure 5.1: Noise Removal and Voronoi++ Illustration.



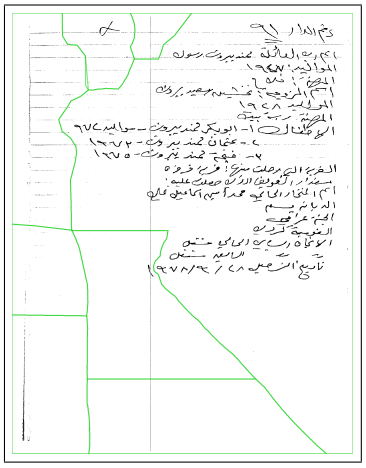
(g) Noisy image



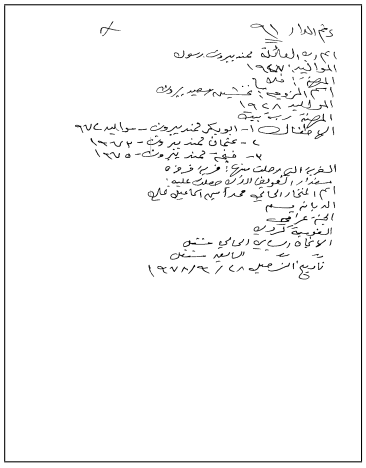
(h) Voronoi++ segmentation on noisy image



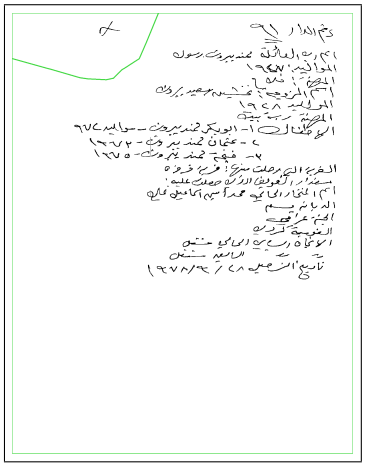
(i) Image with clutter removed



(j) Voronoi++ segmentation on image with clutter removed



(k) Clean image with SPN removed



(l) Voronoi++ segmentation on a clean image

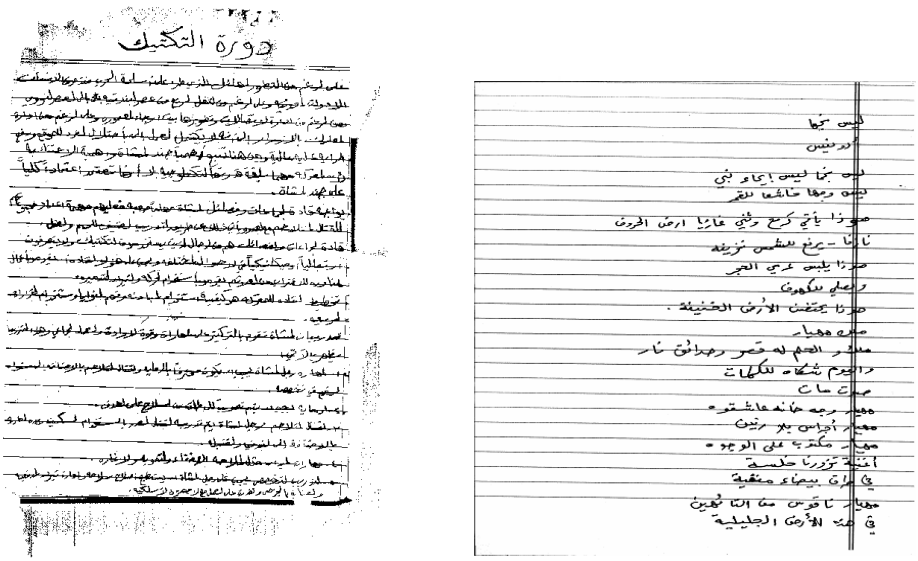
Figure 5.1: Noise Removal and Voronoi++ Illustration.

Another assumption being made in clutter removal is that text always protrudes out from clutter's main body. However, in few cases, it may be possible that the attached text is 'along' the clutter's boundary. In such cases, there may not be a discernible 'knee' in number of regeneration steps as stated in Section 2.2.1.4. This suggests the need for a more sophisticated approach.

2. **Rule-Line Removal:** Figure 5.2a shows the output of the current clutter removal system. It is clear from Figures 5.2a,b that unless character components are obtained from the resultant image, any meaningful interpretation or recognition of text is still challenging. Hence, removal of ruled lines which connect various character components into one, is a necessary goal.

   Instead of a pixel-based approach (which tends to be computationally expensive [9]), a possible enhancement can be done through DSCC (Directional Single Connected Chains [102]) combined with PTC (Prominent Text Components) to remove rule-lines. DSCCs extract the longest sub-components of a rule-line from a connected component of rule-line and text. The challenge is to make the system automatically adapt from each document image, by providing semi-automated cues, in order to avoid the problem of dissimilarity between training and test data [102, 9].

3. **Generic Noise Detection Model:** An individual detection process for each kind of noise can be expensive, and at the same time, due to various shapes and forms of noise, it is difficult to *identify* **NOISE** as a whole, than *reject*



(a) (b)

Figure 5.2: Documents with ruled lines touching character components.

it as **NON-CONTENT**. Training a generic recognizer on any kind of noise may not be possible. Larry and Malik [57] designed a single class classifier which is trained on positive samples only and rejects any sample not in the *trained* class as **OTHER**. Using the same principle, it should be possible to train a single-class SVM on clean document images, which can then reject a document with any kind of noise (non-clean). Due to its independent nature with respect to other types of noise, the clutter removal approach can be combined with this generic noise removal framework. The residual process on a clutter document removes anything with similar or smaller width than text-stroke width. This also removes noise like salt-n-pepper, rule-lines, bleed-through and stray marks, and hence avoids their interference with clutter detection and removal processes.

4. **Prominent Text Components** We would like to extend this approach to

other scripts and documents with mixed content. The idea of separating prominent text components first and using their properties to perform context analysis can be utilized in many other domains of document processing, like rule-line removal, line extraction and word segmentation.

5.2 Page Segmentation and Zone classification

Summary: In Chapter 3 we presented a dynamic approach to content-based Voronoi page segmentation which makes context-aware decisions using both low-level inter-component relationship features and high-level zonal content features. Our approach consists of a *hypothesis* phase followed by a *validation* phase. The *hypothesis* phase creates over-segmented zones based on low-level features, such as component separations and variations, using a dynamically adaptive approach. We then remove inter-component edges only if the corresponding (separated) content is similar and spatially close. In case of conflicts in global or local parameters, edges are left as provisional for further validation. The *validation* phase builds context using these low-level features (distance context) and high-level content features (similarity context). A decision to form zones is then based on a context-aware system which merges zones only if they are similar in both contexts. We compared the Voronoi and Voronoi++ (with and without context) approaches using various zone-based evaluation schemes and showed that the context-aware Voronoi++ performs better than Voronoi and the dynamically adaptive context-unaware system in all the schemes.

Future Directions: To improve performance further, a learning approach may be required. We will attempt to bootstrap the system to learn from its clustering mistakes. The errors made by the semi-supervised clustering algorithm will be fed back to the system for further improvement. This should be a self-learning system which should learn to adapt to separation and similarity contexts. As we introduce more features per zone, optimization approaches will be useful to model a better trained system.

Another possible directions of this research are in zone-based document retrieval, zone detection and document similarity. Document retrieval based on similar zones (e.g. logos, signatures, seals, stamps) does not typically warrant the need of already segmented document images. In fact, a query zone instance can provide cues to segment and extract the regions-of-interest from a database of document images. Voronoi++ *hypothesis*-phase is a conservative step in zone segmentation which produces zones quite unlikely to be split any further. The features extracted from the query zone instance can help the *separation* and *similarity* contexts in the *validation*-phase. A more confident merger of the correctly hypothesized sub-zones (from *hypothesis*-phase) can then be performed, in turn retrieving the document image with the marked zone of interest.

5.3 Character Segmentation and Recognition

Summary: In Chapter 4 we have presented a novel technique to intelligently segment and recognize characters in complex syllabic scripts, using font-models.

The approach emphasizes the importance of a good feature extraction module (directional features over template or Zernike moments). These techniques not only enhance degraded text-recognition results, but also work with a limited number of training samples. An intelligent conjunct-detection scheme was also proposed which is more intuitive than previous approaches. These techniques do not differentiate syllabic or non-syllabic approaches for segmentation and hence carry out direct character segmentation from words even for syllabic scripts. The approach is targeted towards word-based recognition and hence is ready for language-models. This technique however is slower than dissection-based segmentation and recognition, as it requires the analysis of recognition results at every possible segmentation hypotheses. It is also susceptible to mis-recognition if the font-size of a character changes abruptly within a single word. This is because the model works on a self-learning approach by tuning its parameters using the next recognized character.

Future Directions: As illustrated in Section 4.3.3, an intelligent conjunct detection scheme was developed using the combination possibility of two or more characters using font-files. However, in syllabic languages, conjunct characters (or syllables) are generally two-dimensional fusions of basic consonants or vowels. It is not always possible to segment them by a vertical cut and likewise we can not train on all possible conjuncts (or syllables), because:

- A potentially large set of conjuncts exist due to multiplicative combinations of glyphs.

- In low-density languages, insufficient coverage of conjuncts will result from limited ground-truth.

- Atomic units of syllabic scripts are still characters, instead of syllables. Hence, it makes sense to train a classifier on basic units and learn compositional rules between them.

Using the same principle, it should also be possible to create templates of composed characters using trained templates of consonants and vowels and compositional rules from font files.

Composition has been generally argued to be the fundamental to language [36, 67, 18]. Just like objects and scenes can be decomposed into a hierarchy of meaningful and generic parts, each word can be decomposed into a hierarchy of syllables, conjuncts and atomic glyphs. By establishing a compositional representation, the complexity of object models can be reduced and learning such models from limited training data becomes feasible. However, a fundamental concept is to find a trade-off between two extremes: learning high-level objects entails high intra-category variations while very low-level descriptors fail to capture reliable information on the overall object category. For syllabic script recognition, the primitives can be basic consonants and vowel modifiers, and font-file compositional rules can provide us syllables and conjuncts as compositional objects. Using the training data, the prior assignment of probabilities of these compositional candidates can be computed. This is another potential direction of learning low-density syllabic scripts where the basic training units can be atomic characters

and models for conjuncts and syllables can be created as compositional objects.

Appendix A

Summary of Contributions

1. Clutter Noise Removal

    - A distance transform based approach independent of clutter's position, size, shape and connectivity with text.
    - Precise determination of clutter-text boundary leading to clutter removal while preserving the attached text.

2. Stroke-like Pattern Noise Removal

    - Analysis of noise much similar to text and its effects and interaction with foreground content.
    - A two-phased approach in understanding prominent text components with independent features and smaller text components with dependent features.

3. Page Segmentation

    - A context aware and dynamically adaptive approach to document page segmentation based on inter-component relationships, local patterns and context features.

- A two-phased approach where connected components are first combined using their low-level separation features, and later verified based on the context built from local separation and high-level content features.

4. Zone Classification

   - Combination scheme of structural and texture features to represent each region for zone classification.
   - A new hybrid approach to improve classification accuracy instead of classic one-against-all or one-against-one approaches.

5. Character Segmentation and Recognition

   - A generic character-segmentation and recognition scheme for syllabic and non-syllabic languages using font-models.
   - Building a character segmentation and recognition system for low-density languages using a limited training set.

Appendix B

List of Publications

1. Mudit Agrawal and David Doermann. "Voronoi++: Context-Aware Dynamic Page Segmentation," *Pattern Analysis and Machine Intelligence (PAMI)*, 2011. (submitted)

2. Mudit Agrawal and David Doermann. "Clutter Noise Removal in Binary Document Images," *International Journal on Document Analysis and Recognition (IJDAR)*, 2011. (submitted)

3. Mudit Agrawal and David Doermann. "Stroke-like Pattern Noise Removal in Binary Document Images," *International Conference on Document Analysis and Recognition (ICDAR'11)*, 18-21 Sep. 2011.

4. Mudit Agrawal and David Doermann, "Context-Aware and Content-Based Dynamic Voronoi Page Segmentation," *9th IAPR International Workshop on Document Analysis Systems (DAS '10)*, pp. 73-80, Jun. 2010.

5. Wontaek Seo, Mudit Agrawal, David Doermann, "Performance Evaluation Tools for Zone Segmentation and Classification (PETS)," *20th International Conference on Pattern Recognition (ICPR'10)*, pp. 503-506, 23-26 Aug. 2010.

6. Mudit Agrawal and David Doermann, "Clutter Noise Removal in Binary Document Images," *10th International Conference on Document Analysis and*

Recognition (ICDAR'09), pp. 556-560, 26-29 Jul. 2009.

7. Mudit Agrawal and David Doermann, "Voronoi++: A Dynamic Page Segmentation approach based on Voronoi and Docstrum features," *10th International Conference on Document Analysis and Recognition (ICDAR'09)*, pp. 1011-1015, 26-29 Jul. 2009.

8. Mudit Agrawal, Huanfeng Ma and David Doermann. "Chapter: Generalization of Hindi OCR using Adaptive Segmentation and Font Files," *Guide to OCR for Indic Scripts*, Springer, 2009.

9. W. Abd-Almageed, Mudit Agrawal, Wontaek Seo and David Doermann, "Document Zone Classification Using Partial Least Squares and Hybrid Classifiers," *19th International Conference on Pattern Recognition, (ICPR'08)*, pp. 1-4, 8-11 Dec. 2008.

10. Mudit Agrawal and David Doermann, "Re-Targetable OCR with Intelligent Character Segmentation," *8th IAPR International Workshop on Document Analysis Systems (DAS '08)*, pp. 183-190.

Bibliography

[1] Digital library of India. http://www.dli.cdacnoida.in/.

[2] Google book search. http://books.google.com/.

[3] Kb digitization. http://www.kb.nl/hrd/digitalisering/index-en.html.

[4] Million book project. http://www.archive.org/details/millionbooks.

[5] Open content alliance. http://www.opencontentalliance.org/.

[6] *Matlab Image Processing Toolbox: Connected component properties*, 2001. http://www.mathworks.com/help/toolbox/images/ref/regionprops.html.

[7] Mehmmood Abdulla Abd and George Paschos. Effective Arabic character recognition using Support Vector Machines. In *Innovations and Advanced Techniques in Computer and Information Sciences and Engineering*, pages 7–11, London, UK, 2007. Springer-Verlag.

[8] W. Abd Almageed, M. Agrawal, W. Seo, and D. Doermann. Document-zone classification using partial least squares and hybrid classifiers. *Int'l Conf. on Pattern Recognition (ICPR'08)*, pages 1–4, 2008.

[9] W. Abd-Almageed, J. Kumar, and D. Doermann. Page rule-line removal using linear subspaces in monochromatic handwritten arabic documents. In *10th Int'l Conf. on Document Analysis and Recognition (ICDAR'09)*, pages 768 –772, Jul. 2009.

[10] M. Agrawal and D. Doermann. Clutter noise removal in binary document images. In *10th Int'l Conf. on Document Analysis and Recognition (ICDAR'09)*, pages 556 –560, Jul. 2009.

[11] Mudit Agrawal and David Doermann. Voronoi++: A dynamic page segmentation approach based on voronoi and docstrum features. In *Proc. 10th Int'l Conf. on Document Analysis and Recognition (ICDAR'09)*, pages 1011–1015, 2009.

[12] M.B.H. Ali. Background noise detection and cleaning in document images. *Proc. 13th Int'l Conf. Pattern Recognition, (ICPR'96)*, 3:758–762 vol.3, Aug. 1996.

[13] Ronald L. Allen and Duncan Mills. *Signal Analysis: Time, Frequency, Scale, and Structure*. Wiley-IEEE Press, 2004.

[14] A. Antonacopoulos and R.T. Ritchings. Flexible page segmentation using the background. In *Proc. 12th Int'l Conf. on Pattern Recognition (ICPR'94)*, volume 2, pages 339–344, Oct 1994.

[15] Bruno Tenório Ávila and Rafael Dueire Lins. A new algorithm for removing noisy borders from monochromatic documents. In *Proc. of the 2004 ACM Symposium on Applied Computing (SAC '04)*, pages 1219–1225, New York, NY, USA, 2004.

[16] Henry S. Baird. Background structure in document images. In *Advances in Structural and Syntactic Pattern Recognition*, pages 17–34. World Scientific, 1994.

[17] U. Bhattacharya, T.K. Das, A. Datta, S.K. Parui, and B.B. Chaudhuri. A hybrid scheme for handprinted numeral recognition based on a self-organizing network and MLP classifiers. *Int'l Journal for Pattern Recognition and Artificial Intelligence (PAMI)*, 16(7):845–864, 2002.

[18] I. Biederman. Recognition-by-components: A theory of human image understanding. *Psychological Review*, 94:115–147, 1987.

[19] Gunilla Borgefors. Distance transformations in digital images. *Computer Vision Graphics and Image Processing (CVGIP'86)*, 34(3):344–371, 1986.

[20] Thomas M. Breuel. Two geometric algorithms for layout analysis. In *Proc. of the 5th Int'l Workshop on Document Analysis Systems V (DAS'02)*, DAS '02, pages 188–199, London, UK, 2002. Springer-Verlag.

[21] A.S. Britto, R. Sabourin, F. Bortolozzi, and C.Y. Suen. The recognition of handwritten numerals strings using a two-stage HMM based method. *Int'l Journal of Document Analysis and Recognition (IJDAR)*, 5:102–117, 2003.

[22] R. C. Gonzalez and R. E. Woods. *Digital Image Processing*. Addison-Wesley, third edition, 2008.

[23] G. Casey and E.Lecolinet. A survey of methods and strategies in character segmentation. *IEEE Trans. on Pattern Analysis and Machine Intelligence (PAMI)*, 18:690–706, July 1996.

[24] F. Cesarini, M. Lastri, S. Marinai, and G. Soda. Encoding of modified X-Y trees for document classification. In *Int'l Conf. on Document Analysis and Recognition (ICDAR'01)*, pages 1131–1135, 2001.

[25] Chih-Chung Chang and Chih-Jen Lin. *LIBSVM: a library for support vector machines*, 2001. Software available at http://www.csie.ntu.edu.tw/ cjlin/libsvm.

[26] J. Chen, M. K. Leung, and Y. Gao. Noisy logo recognition using line segment Hausdorff distance. *Pattern Recognition*, 36(4):943–955, 2003.

[27] Y. Chi and H. Yan. Handwritten numeral recognition using self-organizing maps and fuzzy rules. *Pattern Recognition*, 28:56–66, 1995.

[28] K. Chinnasarn, Y. Rangsanseri, and P. Thitimajshima. Removing salt-and-pepper noise in text/graphics images. *Asia-Pacific Conf. on Circuits and Systems (IEEE APCCAS)*, pages 459–462, Nov. 1998.

[29] C. Choisy and A. Belaid. Cross-learning in analytic word recognition without segmentation. *Int'l Journal on Document Analysis and Recognition (IJDAR)*, 4:281–289, 2002.

[30] S. P. Chowdhury, S. Mandal, A. K. Das, and Bhabatosh Chanda. Automated segmentation of math-zones from document images. In *Proc. of the 7th Int'l Conf. on Document Analysis and Recognition (ICDAR'03)*, page 755, Washington, DC, USA, 2003. IEEE Computer Society.

[31] D. Dhanya and A. G. Ramakrishnan. Optimal feature extraction for bilingual OCR. In *Proc. of the 5th Int'l Workshop on Document Analysis Systems V (DAS'02)*, pages 25–36, London, UK, 2002. Springer-Verlag.

[32] F. Esposito, D. Malerba, and G. Semeraro. Classification in noisy environments using a distance measure between structural symbolic descriptions. *IEEE Trans. Pattern Analysis and Machine Intelligence (PAMI)*, 14:390–402, 1992.

[33] Kuo-Chin Fan, Yuan-Kai Wang, and Tsann-Ran Lay. Marginal noise removal of document images. *Proc. 6th Int'l Conf. Document Analysis and Recognition (ICDAR'01)*, pages 317–321, 2001.

[34] V. Ferrari, L. Fevrier, F. Jurie, and C. Schmid. Groups of adjacent contour segments for object detection. *IEEE Trans. Pattern Analysis and Machine Intelligence (PAMI)*, 30(1):36–51, 2008.

[35] Lloyd A. Fletcher and Rangachar Kasturi. A robust algorithm for text string separation from mixed text/graphics images. *IEEE Trans. Pattern Analysis Machine Intelligence (PAMI)*, 10(6):910–918, 1988.

[36] S. Geman, D.F. Potter, and Z. Chi. Composition systems. *Quarterly of Applied Mathematics*, 60, 2002.

[37] L. O' Gorman and R. Kasturi. Document Image Analysis: A bibliography. *Machine Vision and Applications*, 5(3):231–243, 1992.

[38] G. H. Granlund. Fourier Preprocessing for Hand Print Character Recognition. *IEEE Trans. on Computers*, C–21(2):195–201, 1972.

[39] Isabelle Guyon, Robert M. Haralick, Jonathan J. Hull, and Ihsin Tsaiyun Phillips. Data sets for ocr and document image understanding research. In *Proc. of SPIE - Document Recognition IV*, pages 779–799. World Scientific, 1997.

[40] F. Hones and J. Lichter. Layout extraction of mixed-mode documents. *Machine Vision Appl.*, 7(4):237–246, 1994.

[41] J. Hu, R. Kashi, D. Lopresti, and G. Wilfong. Evaluating the performance of table processing algorithms. *Int. Journal on Document Analysis and Recognition (IJDAR)*, 4(3):140–153, 2002.

[42] A.K. Jain and Y. Zhong. Page segmentation using texture analysis. *Pattern Recognition*, 29(5):743–770, May 1996.

[43] Anil K. Jain and Sushil Bhattacharjee. Document image analysis. chapter Text segmentation using Gabor filters for automatic document processing, pages 182–197. IEEE Computer Society Press, Los Alamitos, CA, USA, 1995.

[44] Nei Kato, Masato Suzuki, Shin'ichiro Omachi, Hirotomo Aso, and Yoshiaki Nemoto. A handwritten character recognition system using directional element feature and asymmetric mahalanobis distance. *IEEE Trans. Pattern Analysis and Machine Intelligence (PAMI)*, 21(3):258–262, 1999.

[45] A. Khotanzad and Y. H. Hong. Invariant image recognition by Zernike moments. *IEEE Trans. on Pattern Analysis and Machine Intelligence (PAMI)*, 12(5):489–497, 1990.

[46] Koichi Kise, Akinori Sato, and Motoi Iwata. Segmentation of page images using the area voronoi diagram. *Computer Vision and Image Understanding*, 70(3):370–382, 1998.

[47] G.E. Kopec and P.A. Chou. Document image decoding using Markov source models. *IEEE Trans. on Pattern Analysis and Machine Intelligence (PAMI)*, 16(6):602–617, 1994.

[48] Jayant Kumar, Wael Abd-Almageed, Le Kang, and David Doermann. Handwritten arabic text line segmentation using affinity propagation. In *Proc. of the 9th IAPR Int'l Workshop on Document Analysis Systems (DAS'10)*, DAS '10, pages 135–142, 2010.

[49] D. X. Le, G. R. Thoma, and H. Wechsler. Automated borders detection and adaptive segmentation for binary document images. In *Proc. of the Int'l Conf. on Pattern Recognition (ICPR '96)*, volume 3 of *ICPR '96*, pages 737–, Washington, DC, USA, 1996. IEEE Computer Society.

[50] Su Liang, M. Ahmadi, and M. Shridhar. A morphological approach to text string extraction from regular periodic overlapping text/background images. *Proc. IEEE Int'l Conf. Image Processing (ICIP'94)*, 1:144–148 vol.1, Nov 1994.

[51] Ying Liu and S.N. Srihari. Document image binarization based on texture features. *IEEE Trans. on Pattern Analysis and Machine Intelligence (PAMI)*, 19(5):540 –544, May 1997.

[52] David G. Lowe. Distinctive image features from scale-invariant keypoints. *Int'l J. Computer Vision*, 60(2):91–110, 2004.

[53] S. Jaeger M. Roth and D. Doermann. GEDI: Ground truth editor and document interface. In *Summit on Arabic and Chinese Handwriting Recognition*, 2006.

[54] H. Ma and D. Doermann. Adaptive Hindi OCR using generalized Hausdorff image comparison. *ACM Trans. on Asian Language Information Processing*, 26(2):198–213, 2003.

[55] H. Ma and D. Doermann. Adaptive OCR with limited user feedback. *Int'l Conf. on Document Analysis and Recognition (ICDAR'05)*, pages 814–818, 2005.

[56] S.A. Mahmoud. Arabic character recognition using Fourier descriptors and character contour encoding. *Pattern Recognition*, 27:815–824, 1994.

[57] Larry M. Manevitz and Malik Yousef. One-class svms for document classification. *Journal of Machine Learning Research*, 2:139–154, 2002.

[58] Song Mao and Tapas Kanungo. Automatic training of page segmentation algorithms: An optimization approach. In *Proc. of Int'l Conf. on Pattern Recognition (ICPR'00)*, pages 531–534, 2000.

[59] Magdi Mohamed and Paul Gader. Handwritten word recognition using segmentation-free hidden markov modeling and segmentation-based dynamic programming techniques. *IEEE Trans. on Pattern Analysis and Machine Intelligence (PAMI)*, 18(5):548–554, 1996.

[60] George Nagy, Sharad Seth, and Mahesh Viswanathan. A prototype document image analysis system for technical journals. *Computer*, 25(7):10–22, 1992.

[61] Thomas A. Nartker, Stephen V. Rice, and Steven E. Lumos. Software tools and test data for research and testing of page-reading OCR systems. *Document Recognition and Retrieval XII*, 5676:37–47, 2005.

[62] H. Negishi, J. Kato, H. Hase, and T. Watanabe. Character extraction from noisy background for an automatic reference system. *Proc. 5th Int'l Conf. Document Analysis and Recognition (ICDAR'99)*, pages 143–146, Sep. 1999.

[63] N. Normand and C. Viard-Gaudin. A background based adaptive page segmentation algorithm. In *Proc. 3rd Int'l Conf. on Document Analysis and Recognition (ICDAR'95)*, page 138, Washington, DC, USA, 1995. IEEE Computer Society.

[64] L. O'Gorman. The document spectrum for page layout analysis. *IEEE Trans. Pattern Analysis Machine Intelligence (PAMI)*, 15(11):1162–1173, 1993.

[65] L. O'Gorman and R. Kasturi. *Document Image Analysis*. Computer Society Press, 1995.

[66] Timo Ojala, Matti Pietikäinen, and Topi Mäenpää. Multiresolution Gray-Scale and Rotation Invariant Texture Classification with Local Binary Patterns. *IEEE Trans. Pattern Analysis and Machine Intelligence (PAMI)*, 24(7):971–987, 2002.

[67] B. Ommer and J. M. Buhmann. Learning compositional categorization models. *ECCV*, 3951, 2006.

[68] H. Ozawa and T. Nakagawa. A character image enhancement method from characters with various background images. *Proc. 2nd Int'l Conf. Document Analysis and Recognition (ICDAR'93)*, pages 58–61, Oct. 1993.

[69] Theo Pavlidis and Jiangying Zhou. Page segmentation and classification. *Graphical Models and Image Processing (CVGIP)*, 54(6):484–496, 1992.

[70] Tuan D. Pham. Unconstrained logo detection in document images. *Pattern Recognition*, 36(12):3023 – 3025, 2003.

[71] R. Plamondon and S.N. Srihari. On-line and off-line handwritten recognition: a comprehensive survey. *IEEE Trans. on Pattern Analysis and Machine Intelligence (PAMI)*, 22:62–84, 2000.

[72] Thomas Plötz and Gernot A. Fink. Markov models for offline handwriting recognition: a survey. *Int'l Journal on Document Analysis and Recognition*, 12:269–298, November 2009.

[73] Nucharee Premchaiswadi, Sukanya Yimgnagm, and Wichian Premchaiswadi. A scheme for salt and pepper noise reduction and its application for ocr systems. *WSEAS Trans. on Computers*, 9:351–360, Apr. 2010.

[74] A. Rosenfeld and J. L. Pfaltz. Sequential operations in digital picture processing. *Journal of the Assoc. for Comp. Machine*, 13(4):471–494, 1966.

[75] A. Rosenfeld and J. L. Pfaltz. Distance functions on digital pictures. *Pattern Recognition*, 1(1):33–61, 1968.

[76] Iwao Sekita, Ryoichi Mori, Kazuhiko Yamamoto, Hiromitsu Yamada, and Kazuo Toraichi. Feature extraction of handwritten japanese characters by spline functions for relaxation matching. *Pattern Recognition*, 21(1):9–17, 1988.

[77] Wontaek Seo, Mudit Agrawal, and David Doermann. PETS: Performance Evaluation ToolS (software). http://lamp.cfar.umd.edu (Media Group/Research/PETS).

[78] Wontaek Seo, Mudit Agrawal, and David Doermann. Performance evaluation tools for zone segmentation and classification (PETS). In *Proc. of the 20th Int'l Conf. on Pattern Recognition (ICPR'10)*, ICPR '10, pages 503–506, Washington, DC, USA, 2010. IEEE Computer Society.

[79] J. Serra. *Image Analysis and Mathematical Morphology*. Academic Press, third edition, 1983.

[80] F. Shafait and T.M. Breuel. A simple and effective approach for border noise removal from document images. In *IEEE 13th Int'l Multitopic Conf. (INMIC'09)*, pages 1 –5, Dec. 2009.

[81] Faisal Shafait, Daniel Keysers, and Thomas M. Breuel. Performance comparison of six algorithms for page segmentation. In *7th IAPR Workshop on Document Analysis Systems (DAS'06)*, pages 368–379. Springer, 2006.

[82] Zhixin Shi, S. Setlur, and V. Govindaraju. Removing rule-lines from binary handwritten arabic document images using directional local profile. In *20th Int'l Conf. on Pattern Recognition (ICPR'10)*, pages 1916 –1919, Aug. 2010.

[83] Y. Solihin and C.G. Leedham. Integral ratio: a new class of global thresholding techniques for handwriting images. *IEEE Trans. on Pattern Analysis and Machine Intelligence (PAMI)*, 21(8):761 –768, Aug. 1999.

[84] N. Stamatopoulos, B. Gatos, and T. Georgiou. Automatic borders detection of camera document images. In *Proc. 2nd Int'l Workshop Camera-Based Document Analysis and Recognition (CBDAR'07)*, pages 71–78, Sep. 2007.

[85] N. Stamatopoulos, B. Gatos, and T. Georgiou. Page frame detection for double page document images. In *Proc. of the 9th IAPR Int'l Workshop on Document Analysis Systems (DAS'10)*, DAS '10, pages 401–408. ACM, 2010.

[86] D. Stefano, A.D. Cioppa, and A. Marcelli. Handwritten numeral recognition by means of Evolutionary Algorithms. *Int'l Conf. on Document Analysis and Recognition (ICDAR'99)*, 0:804–807, 1999.

[87] G.A. Story, L. O'Gorman, D. Fox, L.L. Schaper, and H.V. Jagadish. The rightpages image-based electronic library for alerting and browsing. *Computer*, 25(9):17 – 26, Sep. 1992.

[88] C. Strouthopoulos, N. Papamarkos, and A. E. Atsalakis. Text extraction in complex color documents. *Pattern Recognition*, 35(8):1743 – 1758, 2002.

[89] C. Strouthopoulos, N. Papamarkos, and C. Chamzas. Identification of text-only areas in mixed-type documents. *Engineering Applications of Artificial Intelligence*, 10(4):387–401, 1997.

[90] Y. Suen, M. Berthod, and S. Mori. Automatic recognition of handprinted characters the state of the art. *Proc. IEEE*, 68(4):469–487, 1980.

[91] M. Teague. Image analysis via the general theory of moments. *Journal of the Optical Society of America*, 70(8):920–930, 1979.

[92] Qian Wang and Chew Lim Tan. Matching of double-sided document images to remove interference. *Proc. Computer Vision and Pattern Recognition (CVPR)*, 1:I–1084–I–1089 vol.1, 2001.

[93] Yalin Wang, Ihsin T. Phillips, and Robert M. Haralick. Document zone content classification and its performance evaluation. *Pattern Recognition*, 39(1):57–73, 2006.

[94] Piotr S. Windyga. Fast impulsive noise removal. In *IEEE Transactions on Image Processing*, volume 10, pages 173–179, Jan 2001.

[95] H. Wold. *Multivariate Analysis*, chapter Estimation of principal components and related models by iterative least squares. New York: Academic Press, 1966.

[96] K. Y. Wong, R. G. Casey, and F. M. Wahl. Document Analysis System. *j-IBM-JRD*, 26(6):647–656, November 1982.

[97] Victor Wu, Raghavan Manmatha, and Edward M. Riseman, Sr. Textfinder: An automatic system to detect and recognize text in images. *IEEE Trans. Pattern Analysis and Machine Intelligence (PAMI)*, 21(11):1224–1229, 1999.

[98] Y. Xiao and Y. Yan. Test Region Extraction in a document image based on the Delaunay tessellation. *Pattern Recognition*, 36(3):799–809, 2003.

[99] Q. Yuan and C.L. Tan. Text extraction from gray scale document images using edge information. In *Proc. 6th Int'l Conf. on Document Analysis and Recognition (ICDAR'01)*, pages 302–306, 2001.

[100] R. Zanibbi, D. Blostein, and J. R. Cordy. Recognizing Mathematical Expressions using Tree Transformation. *IEEE Trans. Pattern Analysis and Machine Intelligence (PAMI)*, 24(11):1455–1467, 2002.

[101] Yefeng Zheng, Huiping Li, and D. Doermann. A model-based line detection algorithm in documents. *Proc. 7th Int'l Conf. on Document Analysis and Recognition (ICDAR'03)*, 1:44–48, Aug. 2003.

[102] Yefeng Zheng, Huiping Li, and David Doermann. A parallel-line detection algorithm based on HMM decoding. *IEEE Trans. Pattern Analysis and Machine Intelligence (PAMI)*, 27(5):777–792, 2005.

[103] Yefeng Zheng, Changsong Liu, Xiaoqing Ding, and Shiyan Pan. Form frame line detection with directional single-connected chain. *Proc. 6th Int'l Conf. Document Analysis and Recognition (ICDAR'01)*, pages 699–703, 2001.

[104] Guangyu Zhu, Stefan Jaeger, and David Doermann. A Robust Stamp Detection Framework on Degraded Documents. In *Int'l Conf. on Document Recognition and Retrieval XIII*, pages 1–9. San Jose, CA, 2006.

[105] Elena Zotkina, Himanshu Suri, and David Doermann. GEDI: Groundtruthing Environment for Document Images (software). http://lamp.cfar.umd.edu (Media Group/Research/GEDI).